```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
```

FILEID**XGDRIVER

```
XX      XX    GGGGGGGG  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX    GGGGGGGG  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX  GG          DD      DD  RR      RR      II      VV      VV  EE          RR      RR
XX      XX  GG          DD      DD  RR      RR      II      VV      VV  EE          RR      RR
  XX  XX    GG          DD      DD  RR      RR      II      VV      VV  EE          RR      RR
  XX  XX    GG          DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    XX      GG          DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
    XX      GG          DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
  XX  XX    GG  GGGGGG  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
  XX  XX    GG  GGGGGG  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
XX      XX  GG      GG  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
XX      XX  GG      GG  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
XX      XX    GGGGGG    DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
XX      XX    GGGGGG    DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
```

```
LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

(5) 806 Standard Driver Tables
(6) 880 P2 buffer verification tables
(10) 1136 CONTROL_INIT - Initialize Sync line device
(11) 1164 UNIT_INIT - Initialize the device unit
(12) 1265 INIT_PARAM, Initialize parameters
(14) 1318 XMITFDT - Transmit I/O FDT routine
(15) 1397 COM_XMITFDT - Common transmit FDT routine
(16) 1487 ALT_ENTRY - Alternate I/O entry
(17) 1605 CLEANFDT/ALT_CLEANFDT - Perform CLEAN FDT processing
(18) 1648 XMITLAPB - Transmit LAPB/BISYNC FDT routine
(19) 1762 RCVFDT - Receive I/O FDT routine
(20) 1853 SETMODEFDT, Set mode I/O operation FDT routine
(21) 1977 SETMODE_CTRL, Perform setmode FDT operation on controller
(22) 2155 ALLOC_PROT_BUFFER, Allocate a buffer to be used by the protocol
(23) 2210 SENSEMODEFDT, Sense Mode I/O operation FDT routine
(24) 2326 SENSEMODE_CTRL, Perform SENSEMODE FDT processing for controller
(25) 2460 SENSE_MODEM - Perform SENSEMODE READ_MODEM FDT processing
(26) 2502 GET_CHAR_BUFS, Get P1 and P2 characteristics buffers
(27) 2574 CHECK_BUFS, Check P1 and P2 buffers for write access
(28) 2627 CHECK_P1, Check P1 buffer address for write access
(29) 2668 ALLOC_P2BUF, Allocate a P2 buffer and charge user's quota
(30) 2737 CHG_UCB, Change UCB parameter values
(31) 2842 SET_DEFAULT_CHAR, Set default characteristics for given protocol
(32) 2896 CHG_TRIB, Change trib parameter values
(33) 2946 STARTIO - Start setmode I/O operation
(34) 3007 START_TRANSMIT - Start transmit I/O
(36) 3102 Load Mapping registers and give to COMBO
(39) 3309 READ_MODEM - Read device modem register
(40) 3353 CLEAN - Abort all outstanding XMT's (All IO's for BISYNC)
(42) 3499 START - Start unit, device and/or protocol
(45) 3711 Unit access routines
(48) 3982 Set protocol characteristic
(49) 4034 START_UNIT_BISYNC - Start up device in BISYNC mode
(50) 4172 START_BISYNC_TIMER - Start up the BISYNC timer
(51) 4215 FILLFREELIST - FILL MESSAGE FREE LIST
(52) 4285 START_RECEIVE_BISYNC - Start receives in BISYNC mode
(53) 4344 START_RECEIVE - Start any receives
(56) 4479 INTERRUPTS
(59) 4747 FORKDONE - Fork process
(60) 4826 FORK_DONE_LAPB
(61) 4889 FORK_DONE_BISYNC - Fork routine for BISYNC mode
(62) 4926 RECEIVE_DONE - Complete a receive buffer
(63) 5034 FINISH_RCV_IO - Finish receive I/O processing
(65) 5116 TRANSMIT_DONE - Transmit completion routine
(66) 5176 TIMER - CTS and Device timer wakeup routines
(67) 5315 BISYNC_TIMER - Bisync timer
(68) 5579 STOP_RECEIVE_BISYNC - Stop the BISYNC mode receiver
(69) 5612 REGDUMP - Error log and diagnostics register dump
(70) 5649 Poke user process on attention condition
(71) 5686 TIMEOUT - TIMEOUT
(72) 5715 STOP_BISYNC_TIMER - Stop the bisync mode timer
(73) 5746 CANCEL - Cancel I/O rotuine
(75) 5794 SHUTDOWN - Shut down unit, device and/or protocol
(78) 6125 VALIDATE_P2, Validate P2 buffer parameters
(78) 6126 VALIDATE_P2_TRIB, Validate P2 buffer with Trib param
(78) 6127 VALIDATE_P2_UCB, Validate P2 buffer with UCB
(79) 6253 UPDATE_P2, Update UCB/TRIB based on P2 buffer parameters
(80) 6328 RETURN_P2, Return UCB/DDCMP buffer parameters
(81) 6388 UNPACK_P2_BUF, Unpack a P2 parameter from P2 buffer

```
0000     1          .TITLE  XGDRIVER - VAX/VMS DMF32 Sync Line Device Driver
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;++
0000    28  ; FACILITY:
0000    29  ;
0000    30  ;       VAX/VMS DMF32 Sync Line Device driver
0000    31  ;
0000    32  ; ABSTRACT:
0000    33  ;
0000    34  ;       This module contains the DMF32 Sync Line driver FDT routines,
0000    35  ;       interrupt dispatcher, interrupt service and fork routines.
0000    36  ;
0000    37  ; AUTHOR:
0000    38  ;
0000    39  ;       Meg Dumont 1-May-81
0000    40  ;
0000    41  ; MODIFIED BY:
0000    42  ;
0000    43  ;       V03-028 MMD0325         Meg Dumont,     24-Aug-1984  16:12
0000    44  ;               Move setting up some of the DDCMP default fields from
0000    45  ;               AWAIT_CONT to ALLOC_PROT_BUFFER. This is a fix for
0000    46  ;               a problem found during powerfail testing.
0000    47  ;
0000    48  ;       V03-027 MMD0321         Meg Dumont,     13-Aug-1984  8:49
0000    49  ;               Fix to SENSEMODE_CTRL where we could get into the  situation
0000    50  ;               of calling DDCMP without makin sure the protocol buffer was
0000    51  ;               allocated. Also a fix to SHUTDOWN_LINE to clear the
0000    52  ;               protocol buffer field in the UCB when the  buffer has been
0000    53  ;               deallocated.
0000    54  ;
0000    55  ;       V03-026 MMD0320         Meg Dumont,     1-Aug-1984  14:07
0000    56  ;               Fix to zero the IRP$L_MEDIA field for all the cases of a
0000    57  ;               SENSEMODE CLEAR/READ count, for PSI and BISYNC. This
```

```
0000     58 ;                  fixes a bug where PSI systems would occasionally crash
0000     59 ;                  if the user tried a SHOW/CLEAR CIRCUIT on a line that
0000     60 ;                  was defined but not turned on.
0000     61 ;
0000     62 ;          V03-025 MMD0307         Meg Dumont,     5-Jul-1984  10:20
0000     63 ;                  Add support for a timeout on transmits in progress. There
0000     64 ;                  is a bug in the DMF32 hardware where it occasionally
0000     65 ;                  loses clocking. If a transmit was inprogress that transmit
0000     66 ;                  hangs. This fix incorporates a timer which times a
0000     67 ;                  transmit and if the transmit doesn't complete in the
0000     68 ;                  specified time than the device is shut down and the
0000     69 ;                  error count is incremented.
0000     70 ;
0000     71 ;                  Fix to check only low byte on XMT errors.
0000     72 ;
0000     73 ;          V03-024 MMD0303         Meg Dumont,     26-Jun-1984  16:26
0000     74 ;                  Fix to test that the protocol buffer has been allocated before
0000     75 ;                  we try to use it in the SENSEMODE and XMIT FDT routines. Also
0000     76 ;                  a fix for PSI the shutdown code where we didn't set up the
0000     77 ;                  protocol buffer address before we jumped to finish the CLEAN
0000     78 ;                  processing.
0000     79 ;
0000     80 ;          V03-023 MMD0298         Meg Dumont,     18-May-1984  10:25
0000     81 ;                  Fix to returning quota on line startup. Add the DEV$M_AVL
0000     82 ;                  bit to UCB$L_DEVCHAR to allow PSI and BISYNC to work.
0000     83 ;
0000     84 ;          V03-022 MMD0295         Meg Dumont,     23-Apr-1984  16:21
0000     85 ;                  Fix to support a new interface with the protocol. This
0000     86 ;                  interface is designed to ensure that retransmit DDCMP messages
0000     87 ;                  will not be done so out of order.
0000     88 ;
0000     89 ;          V03-021 MMD0260         Meg Dumont,     22-Mar-1984  14:35
0000     90 ;                  Fix so that the buffer quota for device isn't returned twice.
0000     91 ;
0000     92 ;          V03-020 MMD0255         Meg Dumont,     28-Feb-1984  17:29
0000     93 ;                  Fixes that make the driver less suspetible to hardware errors.
0000     94 ;
0000     95 ;          V03-019 WHM0001         Bill Matthews   16-Feb-1984
0000     96 ;                  Second part of change to control_init to support
0000     97 ;                  multiple comm devices on one board.
0000     98 ;
0000     99 ;          V03-018 MMD0216         Meg Dumont,     6-Jan-1984  12:56
0000    100 ;                  Fix to SENSEMODE, where we were not returning the
0000    101 ;                  correct value in R0. Fixes a problem which showed
0000    102 ;                  up in the UETP.
0000    103 ;
0000    104 ;          V03-017 MMD0212         Meg Dumont,     9-Dec-1983  12:59
0000    105 ;                  Fix to clear the XG_DS_V_XMTING bit on transmit for
0000    106 ;                  the BISYNC mode, which allows users to transmit a
0000    107 ;                  message after an error has occured. Change to control_init
0000    108 ;                  to support the possiblity of mutlitple dmfs on a given
0000    109 ;                  board.
0000    110 ;
0000    111 ;          V03-016 MMD0210         Meg Dumont,     8-Dec-1983  15:30
0000    112 ;                  New driver on master pack with all of FT2 support including
0000    113 ;                  BISYNC.
0000    114 ;
```

```
0000    115 ;               Fix to the Clean routine to take advantage of a microcode
0000    116 ;               fix where by xmts can be aborted on teh board if the enable
0000    117 ;               bit is cleared. Fix the SENSEMODE fdts' to return the correct
0000    118 ;               status in the IOSB if only a P1 parameter was specified.
0000    119 ;
0000    120 ;       V03-015  These two fixes pertianed to necessary FT1 fixes and are
0000    121 ;       V03-014  incorporated in this driver.
0000    122 ;
0000    123 ;       V03-013 MMD0191         Meg Dumont,     13-Jun-1983  11:33
0000    124 ;               Took out setting the retransmit timer on the circuit. Added
0000    125 ;               a check so that is a buffer is being transmited a time out
0000    126 ;               on the line will not cause the buffer to be lost. Added
0000    127 ;               code to set the xmter on when a time out occurs in
0000    128 ;               DDCMP. This will ensure that the message gets send.
0000    129 ;
0000    130 ;       V03-012 MMD0168         Meg Dumont,     3-May-1983  12:45
0000    131 ;               Take out starting of the DDCMP timer from UNIT_INIT and
0000    132 ;               make it part on the line startup and shutdown via
0000    133 ;               the normal DDCMP interface. ALso include the fix where
0000    134 ;               SETMODE on the controller would fail if a P2 buffer was
0000    135 ;               not included in the QIO.
0000    136 ;
0000    137 ;       V03-011 MMD0165         Meg Dumont,     27-Apr-1983  17:50
0000    138 ;               Fix to change the dropping of DTR from SHUTDOWN_CIRCUIT to
0000    139 ;               SHUTDOWN_LINE. Fix to RECEIVE_INTERRUPT routine to include
0000    140 ;               XG$M_ABORT as a non-fatal receive error.
0000    141 ;
0000    142 ;       V03-010 MMD0013         Meg Dumont,     1-Dec-1982  13:59
0000    143 ;               Fixes for support of HDLC
0000    144 ;                  a) Added support for CLEAN via QIO and enhanaced the CLEAN
0000    145 ;                     code in general
0000    146 ;                  b) Changed the use of NMA$C_LINPR_LAP to NMA$C_LINPR_LAPB
0000    147 ;                  c) Added check for device inactive on LAPB XMT's
0000    148 ;
0000    149 ;       V03-009 MMD0012         Meg Dumont,     13-Oct-1982  12:58
0000    150 ;               Fix problem doing straight QIO's over LAPB.
0000    151 ;
0000    152 ;       V03-008 MMD0011         Meg Dumont,     8-Oct-1982  18:45
0000    153 ;               Added LAPB support. ALso a fix for the problem running the
0000    154 ;               DMF32 in half duplex over real modems. Force the line to
0000    155 ;               wait at least 10 milliseconds before trying to transmit
0000    156 ;               a message after it has received control of the line.
0000    157 ;
0000    158 ;       V03-007 MMD0010         Meg Dumont,     7-Jul-1982  13:25
0000    159 ;               Add $VADEF to System definitions.
0000    160 ;
0000    161 ;       V03-006 MMD0009         Meg Dumont,     23-Jun-1982  13:40
0000    162 ;               Add check to ensure that all outstanding XMT's have been
0000    163 ;               sent before processing any RCV'd messages when the device
0000    164 ;               is running half duplex or trib modes. Fixed a bug in
0000    165 ;               CTS_TIMER where a message could get sent even when XMt'r
0000    166 ;               was active.
0000    167 ;
0000    168 ;       V03-005 MMD0008         Meg Dumont,     18-Jun-1982  17:01
0000    169 ;               Fix a bug in POKE_USER, and Change where the TFB address
0000    170 ;               gets loaded in RECEIVE_DONE
0000    171 ;
```

```
                                ;
                                ;       V03-004 MMD0007         Meg Dumont,     2-Jun-1982  9:57
                                ;               Add conditional code to clear fields in UCB for ERR$$$.
                                ;
                                ;       V03-003 MMD0006         Meg Dumont,     25-May-1982  16:05
                                ;               Fix to sync to IPL$_TIMER before queueing entires onto the
                                ;               timer queue. Fix to CONTROL_INIT wrong register was being
                                ;               used. Added some conditional code for journaling IRP's
                                ;               and some to do extra error counting.
                                ;
                                ;       V03-002 MMD0005         Meg Dumont,     23-Apr-1982  14:03
                                ;               Added a TQE which times CTS coming high for half duplex and
                                ;               multipoint cases. Also added a clear on UCB$L_DEVDEPEND for
                                ;               start up on line and circuit.
                                ;
                                ;       V03-001 MMD0004         Meg Dumont,     23-Mar-1982  13:21
                                ;               Fix to use executive routine to PROBE user P2 buffer. Deleted
                                ;               work around code for micro code bugs. Fix to set transmitter
                                ;               state to IDLE on circuit shutdown.
                                ;
                                ;       V03-003 MMD0003         Meg Dumont,             11-Feb-1982  10:51
                                ;               in trib mode do not time out waiting for CTS to come back.
                                ;               Add fields to count latency errors. Add support for
                                ;               retransmit timer as a line parameter.
                                ;
                                ;       V03-002 MMD0002         Meg Dumont,             5-Feb-1982  15:23
                                ;               Fix bug in the Sensemode read parameters.
                                ;
                                ;       V03-001 MMD0001         Meg Dumont,             2-Feb-1982  13:37
                                ;               Add a check for Maxium Receive Buffer parameter. And fixed
                                ;               a couple problems with Clear_Counters.
                                ;--
```

```
0000    204 ;
0000    205 ; System definitions
0000    206 ;
0000    207         $ACBDEF                                 ; AST control block
0000    208         $BISYNCDEF                              ; Struct for BISYNC protocol buffer
0000    209         $CRBDEF                                 ; Controller request block
0000    210         $CXBDEF                                 ; Complex buffer block
0000    211         $DDBDEF                                 ; Device data block
0000    212         $DDCMPDEF                               ; Constant def's for DDCMP
0000    213         $DEVDEF                                 ; Device charateristics
0000    214         $DLADEF                                 ; Driver/protocol command defs
0000    215         $DLKDEF                                 ; Driver/protocol command defs
0000    216         $DYNDEF                                 ; Dynamic data structures
0000    217         $DPTDEF                                 ; Driver prologue table
0000    218         $GFDEF                                  ; Global field definitions
0000    219         $IDBDEF                                 ; Interrupt data block
0000    220         $IODEF                                  ; I/O function codes
0000    221         $IPLDEF                                 ; Interrupt priority levels
0000    222         $IRPDEF                                 ; I/O packets
0000    223         $JIBDEF                                 ; Job information block
0000    224         $LAPBDEF                                ; Lapb buffer def's
0000    225         $MFDDEF                                 ; Message field descriptor
0000    226         $NMADEF                                 ; Network management def's
0000    227         $PCBDEF                                 ; Process control block
0000    228         $PRDEF                                  ; Processor registers
0000    229         $PRVDEF                                 ; Priv def's
0000    230         $SSDEF                                  ; System service status
0000    231         $TFDEF                                  ; DDCMP buffer def's
0000    232         $TQEDEF                                 ; Timer queue element def's
0000    233         $UBADEF                                 ; UNIBUS adapter registers
0000    234         $UCBDEF                                 ; Unit control block
0000    235         $XGDEF                                  ; XGDRIVER symbols
0000    236         $XMDEF                                  ; XMDRIVER symbols
0000    237         $XMTQDEF                                ; Transmit Q def's
0000    238         $VADEF                                  ; Virtual address defs
0000    239         $VECDEF                                 ; Interrupt vector
0000    240
0000    241 ;
0000    242 ; Local macros
0000    243 ;
0000    244         .MACRO  SETBIT  POS,BAS,?L              ; Set a single bit
0000    245                 BBSS    POS,BAS,L
0000    246         L:
0000    247         .ENDM   SETBIT
0000    248
0000    249         .MACRO  CLRBIT  POS,BAS,?L              ; Clear a single bit
0000    250                 BBCC    POS,BAS,L
0000    251         L:
0000    252         .ENDM   CLRBIT
0000    253
0000    254         .MACRO  PARAM   TYPE,OFFSET,WIDTH,MIN,MAX,INVALID,BASE
0000    255 ;
0000    256 ; Inputs:
0000    257 ;
0000    258 ;       TYPE = Parameter type
0000    259 ;       OFFSET = Offset in the data structure to current value
0000    260 ;       WIDTH = Width of field in the data structure (B,W,L)
```

```
0000    261 ;       MIN = Minimum value parameter is allowed to take
0000    262 ;       MAX = Maximum value parameter is allowed to take
0000    263 ;       INVALID = Invalid flags in status word
0000    264 ;       BASE = Data base (LINE,TRIB)
0000    265 ;
0000    266         .IF BLANK TYPE
0000    267         .WORD   0
0000    268         .IF FALSE
0000    269         $$$TYP = TYPE & PRM_M_TYPE                  ; Isolate type code
0000    270         .IIF NOT_BLANK  <MIN>, $$$TYP = $$$TYP!PRM_M_MIN
0000    271         .IIF NOT_BLANK  <MAX>, $$$TYP = $$$TYP!PRM_M_MAX
0000    272         .IIF NOT_BLANK  <INVALID>, $$$TYP = $$$TYP!PRM_M_INVALID
0000    273                 .WORD   $$$TYP
0000    274         $$$OFF = OFFSET & OFF_M_VALUE               ; Isolate offset only
0000    275         $$$WID = 0                                  ; Set null width
0000    276         .IIF IDN <WIDTH><B>, $$$WID = <1@OFF_V_WIDTH>
0000    277         .IIF IDN <WIDTH><W>, $$$WID = <2@OFF_V_WIDTH>
0000    278         .IIF IDN <WIDTH><L>, $$$WID = <3@OFF_V_WIDTH>
0000    279                 .WORD   $$$OFF!$$$WID
0000    280         .IIF NOT_BLANK  <MIN>, .WORD    MIN
0000    281         .IIF NOT_BLANK  <MAX>, .WORD    MAX
0000    282         .IIF NOT_BLANK  <INVALID>, .WORD        INVALID
0000    283         'BASE'_PRM_BUFSIZ = 'BASE'_PRM_BUFSIZ + 6
0000    284         .ENDC
0000    285         .ENDM   PARAM
0000    286
0000    287         .MACRO  SKIP    BIT,LOC,REG,CONTEXT=W,?L        ; SKIP FIELD
0000    288         BBC     #BIT,LOC,L                  ; Br if field not present
0000    289         TST'CONTEXT     (REG)+              ; Skip next field
0000    290 L:
0000    291         .ENDM   SKIP
```

```
0000    293
0000    294 ;
0000    295 ; This macro translates into the CASEx instruction.  It calculates the
0000    296 ; "base" and "limit" parameters from the <index,displacement> list
0000    297 ; specfied in the 'vector' parameter.  The dispatch table is set up
0000    298 ; such that any unspecified index value within the bounds of the
0000    299 ; transfer vector is associated with a diplacement which transfers
0000    300 ; control to the first location after the CASE statement, i.e., behaves
0000    301 ; as if the index were out of bounds.
0000    302 ;
0000    303 ; Example:
0000    304 ;       $DISPATCH       R0,<-                   ; Message type in R0
0000    305 ;
0000    306 ;               ;index  displacement
0000    307 ;
0000    308 ;               <CI,    NSP$RCV_CI>,-           ; Process CI message
0000    309 ;               <CC,    NSP$RCV_CC>,-           ; Process CC message
0000    310 ;               <DI,    NSP$RCV_DI>,-           ; Process DI message
0000    311 ;               <DC,    NSP$RCV_DI>,-           ; Process DC message
0000    312 ;       >
0000    313 ;       BRW     NSP$RCV_ILLMSG                  ; Message type unknown
0000    314 ;
0000    315 .MACRO  $DISPATCH,      INDX,VECTOR,TYPE=W,NMODE=S^#,?MN,?MX,?S,?SS,?ZZ
0000    316 SS:
0000    317
0000    318         .MACRO  $DSP1,$DSP1_1
0000    319                 .IRP    $DSP1_2,$DSP1_1
0000    320                         $DSP2   $DSP1_2
0000    321                 .ENDR
0000    322         .ENDM
0000    323
0000    324         .MACRO  $DSP2,$DSP2_1,$DSP2_2
0000    325                 .=<$DSP2_1-MN>*2 + 5
0000    326                 .WORD   $DSP2_2-S
0000    327         .ENDM
0000    328
0000    329
0000    330         .MACRO  $BND1,$BND1_1,$BND1_2,$BND1_3
0000    331                 $BND2   $BND1_1,$BND1_2
0000    332         .ENDM
0000    333
0000    334         .MACRO  $BND2,$BND2_1,$BND2_2
0000    335                 .IIF    $BND2_1,$BND2_2-.,      .=$BND2_2
0000    336         .ENDM
0000    337
0000    338         .MACRO  $BND    $BND_1,$BND_2
0000    339                 .IRP    $BND_3,<$BND_2>
0000    340                         $BNDT   $BND_1,$BND_3
0000    341                 .ENDR
0000    342         .ENDM
0000    343
0000    344         .=0
0000    345 ZZ:
0000    346         $BND    GT,<VECTOR>
0000    347 MX:
0000    348         $BND    LT,<VECTOR>
0000    349 MN:
```

```
                     0000   350          .=SS
                     0000   351
                     0000   352 CASE'TYPE       INDX,#<MN-ZZ>,NMODE'<MX-MN>
                     0000   353 S:
                     0000   354          .REPT   MX-MN+1
                     0000   355          .WORD   <MX-MN>*2 + 2
                     0000   356          .ENDR
                     0000   357
                     0000   358          .=S
                     0000   359
                     0000   360          $DSP1   <<VECTOR>>
                     0000   361
                     0000   362          .=<MX-MN>*2 + S + 2
                     0000   363
                     0000   364 .ENDM
                     0000   365
                     0000   366 ;        JNX$$$ = 1                          ; Define for Journalling IRP's
                     0000   367 ;        CTS$$$ = 1                          ; Define for CTS debug
                     0000   368 ;        ERR$$$ = 1                          ; Define for fatal error logging
                     0000   369 ;        DBG$$$ = 1                          ; Define for debug ver of driver
                     0000   370 ;        BISYNC$$$ = 1                       ; Define for version of driver
                     0000   371                                             ; to test out BISYNC
                     0000   372
                     0000   373 ;
                     0000   374 ; Local symbol definitions
                     0000   375 ;
                     0000   376 ; The following symbols are defined to work with the various CASE statements
                     0000   377 ; used within this driver when different actions must be taken according
                     0000   378 ; to protocol type. These are specified so tht the DDMCP case always falls
                     0000   379 ; thru the CASE.
                     0000   380
00000000             0000   381 XG$C_PROTYPE_LAPB = 0
00000001             0000   382 XG$C_PROTYPE_BISYNC = 1
00000002             0000   383 XG$C_PROTYPE_DDCMP = 2
                     0000   384
000186A0             0000   385 XG$C_CTS_DELTA  = 10*1000*10                 ; Delta for CTS timer
                     0000   386                                             ; 10 milli seconds
0003D090             0000   387 XG$C_BISYNC_DELTA = 25*1000*10              ; Delta for Bisync TQE
                     000C   388                                             ; 25 milliseconds
00000008             0000   389 XG$C_BPC_DEFAULT = 8
0000012C             0000   390 XG$C_BRG_DEFAULT = 300
00000100             0000   391 XG_DEF_BUFSIZ   = 256                       ; Default buffer size
0000007F             0000   392 XG$C_SIZEOFQ    = DDCMP$C_SIZEOFQ
00000006             0000   393 XG$C_HEADER     = DDCMP$C_HEADER
FFF3F300             0000   394 XG$C_LINE_PAR   = ^X<FFF3F300>               ; Bits to clear in DEVDEPEND
                     0000   395                                             ; on start of line
FFFFF700             0000   396 XG$C_CIR_PAR    = ^X<FFFFF700>              ; Bits to clear in DEVDEPEND
                     0000   397                                             ; on start circuit
00000000             0000   398 XG$C_IDLE       = 0                         ; State of transmitter "Off"
00000001             0000   399 XG$C_XMTING     = 1                         ; State of transmitter "On"
00000002             0000   400 XG$C_WFCTS      = 2                         ; State of transmitter "Waiting
                     0000   401                                             ; for clear to send"
00000004             0000   402 XG$C_SWFCTS     = 4                         ; Short wait CTS to come high
00000006             0000   403 XG$C_DRPCTS     = 6                         ; Drop RTS and wait for CTS to
                     0000   404                                             ; go away
00000006             0000   405 XG$C_WFCTS_SEC  = 6                         ; # of ticks to wait for CTS
                     0000   406
```

```
00000004  0000   407 DSC$A_POINTER   = 4                          ; Descriptor buffer address
          0000   408
FFFFFFFC  0000   409 DMF_CSR = -4                                 ; DMF offset to Sync line CSR
00000004  0000   410 INT_VEC = 4
00000004  0000   411 NUM_MAP_REG = 4                              ; Number of mapping registers
          0000   412                                              ; to allocate
00000004  0000   413 MAX_RCVS = 4                                 ; Max number of RCVS before
          0000   414
          0000   415 ;
          0000   416 ; $QIO parameter offsets
          0000   417 ;
00000000  0000   418 P1       = 0                                 ; Parameter 1
00000004  0000   419 P2       = 4                                 ; Parameter 2
00000008  0000   420 P3       = 8                                 ; Parameter 3
0000000C  0000   421 P4       = 12                                ; Parameter 4
00000010  0000   422 P5       = 16                                ; Parameter 5
          0000   423
          0000   424                                              ; a transmit must happen
          0000   425 ;
          0000   426 ; Overlay of IRP
          0000   427 ;
          0000   428
          0000   429         $DEFINI IRP
          0000   430
00000021  0000   431 . = IRP$W_FUNC+1                             ; Overlay function word
          0021   432
          0021   433 $DEF     IRP$B_XGFUNC     .BLKB   1          ; DMF driver internal func code
          0022   434
00000040  0022   435 . = IRP$Q_STATION
          0040   436
          0040   437 $DEF IRP$W_QUOTA         .BLKW   1          ; DMF driver byte quota needed
          0042   438
          0042   439 ; Define driver internal function codes stored in IRP$B_XGFUNC of IRP.
          0042   440 ; NOTE:  These are not used as bit offsets, but as values.
          0042   441 ;
          0042   442         _VIELD  XG_FC,0,<-                  ; Internal function codes
          0042   443                 <STRT_CIR>,-                ; Start a tributary
          0042   444                 <STRT_LIN>,-                ; Init the device inc UCB
          0042   445                 <STOP_CIR>,-                ; Stop a tributary
          0042   446                 <STOP_LIN>,-                ; Stop the device (inc stopping
          0042   447                 -                           ; the trib as well)
          0042   448                 <CLEAN>,-                   ; Abort all outstanding XMT's
          0042   449                 <READ_MODEM>,-              ; Read modem register
          0042   450                 >
          0042   451
          0042   452         $DEFEND IRP                         ; End of IRP overlays
          0000   453
          0000   454 ;
          0000   455 ; XGDRIVER UCB extensions
          0000   456 ;
          0000   457         $DEFINI
00000090  0000   458                           .BLKB   UCB$C_LENGTH
          0090   459 $DEF     UCB$L_XG_AST     .BLKL   1          ; Attention AST list
          0094   460 $DEF     UCB$L_XG_TQE     .BLKL   15         ; CTS TQE block
          00D0   461 $DEF     UCB$A_XG_PRO_BUFFER .BLKL       1  ; Place to store the address of
          00D4   462                                             ; the buffer allocated for prot
          00D4   463 $DEF     UCB$Q_XG_ATTN    .BLKQ   1          ; Received message list
```

```
                00DC   464 $DEF     UCB$Q_XG_RCVS    .BLKQ   1              ; Receive I/O packet queue
                00E4   465 $DEF     UCB$Q_XG_FREE    .BLKQ   1              ; Free receive buffer queue
000000EC        00EC   466          UCB$L_XG_RCV_INPR =.                    ; Rcv buffer awaiting data for BISYN
000000F0        00EC   467          UCB$W_XG_RCV_INPR_INDX =.+4             ; Index into the rcv buffer
000000F2        00EC   468          UCB$W_XG_RCV_INIT =.+6                  ; Initial value of rcv byte count
                00EC   469 $DEF     UCB$Q_XG_RCV_INPR .BLKQ 1               ; Rcv buffer awaiting data
                00F4   470 $DEF     UCB$Q_XG_POST    .BLKQ 1                ; Receive buffers to I/O comp
                00FC   471 $DEF     UCB$Z_XG_XMT_INPR .BLKQ 1               ; Inpr list for XMT's
                0104   472 $DEF     UCB$L_XG_XMTEND .BLKL   1               ; Timeout time for xmt inpr
                0108   473 $DEF     UCB$L_XG_XMT_TIMEOUT .BLKL 1            ; Length of timeout (in seconds)
                010C   474
                010C   475 $DEF     UCB$W_XG_QUOTA   .BLKW   1              ; Byte quota for RCV's
                010E   476 $DEF     UCB$B_XG_INUS    .BLKB   1              ; Bit set for each slot in use
                010F   477 $DEF     UCB$B_XG_COUNT   .BLKB   1              ; Limit the number of RCVs
                0110   478                                                  ; before a transmit must happen
                0110   479 $DEF     UCB$Z_XG_VECTOR .BLKL   4               ; Control slot vector XMT
                0120   480                                                  ; slot 0 and 1 RCV slot 2 and 3
                0120   481 $DEF     UCB$B_XG_XSTATE .BLKB   1               ; Transmitter state
                0121   482 $DEF     UCB$B_XG_WFCTS_SEC .BLKB 1              ; Number of seconds to wait for
                0122   483                                                  ; clear to send to come up
                0122   484 $DEF     UCB$W_XG_RCVCSR .BLKW   1               ; Last RCV csr value
                0124   485 $DEF     UCB$W_XG_XMTCSR .BLKW   1               ; Last XMT csr value
                0126   486 $DEF     UCB$W_XG_RCVERR .BLKW   1               ; Last RCV error value
                0128   487 $DEF     UCB$W_XG_XMTERR .BLKW   1               ; Last XMT error value
                012A   488 $DEF     UCB$W_XG_DSC     .BLKW   1              ; Last data set change csr
                012C   489
                012C   490 $DEF     UCB$L_XG_PID     .BLKL   1              ; Process ID
                0130   491 $DEF     UCB$W_XG_CHANL   .BLKW   1              ; Line channel number
                0132   492 $DEF     UCB$W_XG_CHANC   .BLKW   1              ; Circuit channel number
                0134   493
                0134   494 $DEF     UCB$Z_XG_DDCMP   .BLKL   4              ; Block for setable DDCMP param
                0144   495 $DEF     UCB$Z_XG_SYNC    .BLKL   3              ; Block for setable SYNC params
                0150   496 $DEF     UCB$B_XG_SETPRM                         ; Start of UCB/line params
                0150   497 $DEF     UCB$B_XG_PRO     .BLKB   1              ; Protocol selection
                0151   498 $DEF     UCB$B_XG_DUP     .BLKB   1              ; Duplex setting
                0152   499 $DEF     UCB$B_XG_CON     .BLKB   1              ; Controller loopback
                0153   500 $DEF     UCB$B_XG_BFN     .BLKB   1              ; Number of receive buffers
                0154   501 $DEF     UCB$B_XG_SPD     .BLKB   1              ; Set the line speed
                0155   502 $DEF     UCB$B_XG_TIMEOUT .BLKB   1              ; Length of CTS timeout
                0156   503 $DEF     UCB$B_XG_MODEM_CLR .BLKB 1              ; Determines modem status after deas
                0157   504 $DEF     UCB$B_XG_MNT_LOOPB .BLKB 1              ; Tyoe of loopb wanted
                0158   505 $DEF     UCB$A_XG_FRAME_ADDR .BLKL 1             ; Address of framing routine passed
0000015C        015C   506          UCB$L_XG_STATE_INFO1 =.                 ; used by framing routine
00000160        015C   507          UCB$L_XG_STATE_INFO2 =.+4
                015C   508 $DEF     UCB$Q_XG_STATE_INFO .BLKQ 1             ; Quadword of BISYNC state info
                0164   509 $DEF     UCB$B_XG_XMTCNT .BLKB   1               ; Number of outstanding XMT's
                0165   510 $DEF     UCB$W_XG_MFDLEN .BLKW   1               ; Size of protocol header
                0167   511 $DEF     UCB$B_XG_PROTYPE .BLKB  1               ; Protocol type used for CASE's
                0168   512 $DEF     UCB$A_XG_CLEAN   .BLKL   1              ; Field for CLEAN IRP address
                016C   513 $DEF     UCB$Z_XG_DLA_ADDR .BLKB DLA$C_ADDR_LENGTH ; Buffer to pass to protocol
                0174   514                                                  ; various address prot needs
                0174   515                                                  ; for DDCMP retransmit of messages
                0174   516
                0174   517 ; Notice default for CHAR is Full duplex control station on a sync line
                0174   518
                0174   519 ;$DEF   TF$A_TFB                                ; Start of trib control block
                0174   520 ; $EQU  TF$V_STATYP     0              ; 0 = Control          1 = Tributary
```

```
                                                H 3
0174    521 ; $EQU  TF$M_STATYP     1
0174    522 ; $EQU  TF$V_POINT      1               ; 0 = Point to point    1 = Multipoint
0174    523 ; $EQU  TF$M_POINT      2
0174    524 ; $EQU  TF$V_DUPLEX     2               ; 0 = Full duplex       1 = Half dplx
0174    525 ; $EQU  TF$M_DUPLEX     4
0174    526 ; $EQU  TF$V_LNTYP      3               ; 0 = Synchronous       1 = Async
0174    527 ; $EQU  TF$M_LNTYP      8
0174    528 ; $DEF  TF$B_CHAR       .BLKB           ; Device characteristics
0174    529 ; $DEF  TF$B_ADDR       .BLKB           ; Station address, default is 1
0174    530 ; $DEF  TF$B_XADDR      .BLKB           ; XMT station address default is 1
0174    531 ; $DEF  TF$B_RADDR      .BLKB           ; RCV stattion address default is 1
0174    532 ; $EQU  TF$V_SNAK       0               ; Send a NAK
0174    533 ; $EQU  TF$M_SNAK       1
0174    534 ; $EQU  TF$V_SACK       1               ; Send an ACK
0174    535 ; $EQU  TF$M_SACK       2
0174    536 ; $EQU  TF$V_SREP       2               ; Send a REP
0174    537 ; $EQU  TF$M_SREP       4
0174    538 ;
0174    539 ;; Message exchange fields
0174    540 ;
0174    541 ; $DEF  TF$B_SFLAGS     .BLKB           ; Send an ENQ message flags
0174    542 ; $DEF  TF$B_R  .BLKB                   ; Highest sequential msg RCV'd
0174    543 ; $DEF  TF$B_N  .BLKB                   ; Highest sequential msg XMT'd
0174    544 ; $DEF  TF$B_A  .BLKB                   ; Highest sequential msg ACK'd
0174    545 ; $DEF  TF$B_T  .BLKB                   ; Next data msg to XMT
0174    546 ; $DEF  TF$B_X  .BLKB                   ; Last data msg to be XMT'd
0174    547 ;
0174    548 ;; Timers
0174    549 ;
0174    550 ; $EQU  TF$V_SELECT     0               ; When set then send the select flag
0174    551 ; $EQU  TF$M_SELECT     1
0174    552 ; $EQU  TF$V_TIMER      1               ; If set then the timer is running
0174    553 ; $EQU  TF$M_TIMER      2
0174    554 ; $EQU  TF$V_RCVDET     2               ; Set when a msg is rcv'd used to
0174    555 ; $EQU  TF$M_RCVDET     4               ; determine if a seletion has been
0174    556 ;                                       ; acknowledged in some way
0174    557 ; $DEF  TF$B_SELTIM     .BLKB           ; Selection and timer flags
0174    558 ; $DEF  TF$B_REPTIM     .BLKB           ; Type of reply timer
0174    559 ; $DEF  TF$W_REPWAI     .BLKW           ; Amount of time to wait before timing
0174    560 ;                                       ; the reply timer out. This field is
0174    561 ;                                       ; only used for timing via a real clock
0174    562 ; $DEF  TF$B_MAXRTO     .BLKB           ; Max number of reply timeouts allowed
0174    563 ; $DEF  TF$B_CURRTO     .BLKB           ; Current number of reply timeouts
0174    564 ; $DEF  TF$B_MSGCNT     .BLKB           ; Max number of messages allowed to
0174    565 ;                                       ; send in one selection interval
0174    566 ; $DEF  TF$B_MMCTR      .BLKB           ; Counter of these msgs
0174    567 ; $DEF  TF$W_TEB        .BLKW           ; Size of trib error block
0174    568 ; $DEF  TF$Q_XMTQ       .BLKQ           ; List head for the XMTQ
0174    569 ; $DEF  TF$Q_CTLQ       .BLKQ           ; List head for control message queue
0174    570 ; $DEF  TF$Q_RTOQ       .BLKQ           ; List head for the RTOQ
0174    571 ; $DEF  TF$Q_CMPQ       .BLKQ           ; List head for the CMPQ
0174    572 ; $DEF  TF$Q_XMTOVF     .BLKQ           ; List head for XMt over flow queue
0174    573 ; $DEF  TF$A_BUFPTR     .BLKL           ; This field gets the value of
0174    574 ;                                       ; R8 on entry to the protocol
0174    575 ; $DEF  TF$A_GFB        .BLKL           ; Address of the global field block
0174    576 ; $DEF  TF$A_DEV_TIMER  .BLKL           ; Address of the device timer routine
0174    577 ;
```

```
0174    578 ; Xmit queue blocks for the control messages
0174    579 ;
0174    580 ; $DEF   TF$L_QACK       .BLKL   12      ; Queue block for an ACK message
0174    581 ; $DEF   TF$L_QNAK       .BLKL   12      ; Queue block for an NAK message
0174    582 ; $DEF   TF$L_QREP       .BLKL   12      ; Queue block for an REP message
0174    583 ; $DEF   TF$L_QSTRT      .BLKL   12      ; Queue block for ar STRT message
0174    584 ; $DEF   TF$L_QSTACK     .BLKL   12      ; Queue block for an STACK message
0174    585 ;
0174    586 ; $DEF   TF$L_TQE        .BLKL   15      ; Timer queue entry block
0174    587 ;
0174    588 ; $DEF   TF$B_NAKRSN     .BLKB           ; NAK reason code
0174    589 ; $DEF   TF$B_XQCNT      .BLKB           ; Count of number of free slots on XMTQ
0174    590 ;
0174    591 ;; Beginning of error counters
0174    592 ;
0174    593 ; $DEF   TF$K_ERRSTRT                   ; Start of error counters
0174    594 ; $DEF   TF$W_DBRTYP     .BLKW           ; NMA definitions for the field
0174    595 ; $DEF   TF$L_DBYTR      .BLKL           ; Records bytes RCV'd by station
0174    596 ; $DEF   TF$W_DBXTYP     .BLKW           ; NMA definitions for the field
0174    597 ; $DEF   TF$L_DBYTX      .BLKL           ; Records bytes XMT'd by station
0174    598 ; $DEF   TF$W_DMRTYP     .BLKW           ; NMA definitions for the field
0174    599 ; $DEF   TF$L_DMSGR      .BLKL           ; Records msgs RCV'd by station
0174    600 ; $DEF   TF$W_DMXTYP     .BLKW           ; NMA definitions for the field
0174    601 ; $DEF   TF$L_DMSGX      .BLKL           ; Records msgs XMT'd by station
0174    602 ; $DEF   TF$W_SELTYP     .BLKW           ; NMA definitions for the field
0174    603 ; $DEF   TF$W_SELSP      .BLKW           ; Records selection interval expired
0174    604 ; $DEF   TF$W_DEOTYP     .BLKW           ; NMA def for field
0174    605 ; $DEF   TF$W_DEOBC      .BLKW           ; Data errors outbound bit counters
0174    606 ; $EQU   TF$V_OHCRC      0               ; NAK's RCV'd header CRC reason code 1
0174    607 ; $EQU   TF$M_OHCRC      1
0174    608 ; $EQU   TF$V_ODCRC      1               ; NAK's RCV'd data CRC reason code 2
0174    609 ; $EQU   TF$M_ODCRC      2
0174    610 ; $EQU   TF$V_OREPS      2               ; NAK's RCV'd REP response rsn code 3
0174    611 ; $EQU   TF$M_OREPS      4
0174    612 ; $DEF   TF$B_DEO        .BLKB           ; Data errors outbound
0174    613 ;        UCB$C_XG_LAPBCTR =.             ; Start of LAPB error ctrs
0174    614 ; $DEF   TF$W_DEITYP     .BLKW           ; NMA def for fields
0174    615 ; $DEF   TF$W_DEIBC      .BLKW           ; Data error inbound error counters
0174    616 ; $EQU   TF$V_IHCRC      0               ; NAK's XMT'd header CRC reason code 1
0174    617 ; $EQU   TF$M_IHCRC      1
0174    618 ; $EQU   TF$V_IDCRC      1               ; NAK's XMT'd data CRC reason code 2
0174    619 ; $EQU   TF$M_IDCRC      2
0174    620 ; $EQU   TF$V_IREPS      2               ; NAK's XMT'd REP response rsn code 3
0174    621 ; $EQU   TF$M_IREPS      4
0174    622 ; $DEF   TF$B_DEI        .BLKB           ; Data error inbound
0174    623 ;        UCB$C_XG_LAPBCTR_LEN =.-UCB$C_XG_LAPBCTR ; Size of error ctrs
0174    624 ; $DEF   TF$W_LBETYP     .BLKW           ; NMA definition for the field
0174    625 ; $DEF   TF$W_LBEBC      .BLKW           ; Local buffer error bit counters
0174    626 ; $EQU   TF$V_LBUF_NAVL  0               ; Local buffer unavl SNAK set rsn 8
0174    627 ; $EQU   TF$M_LBUF_NAVL  1
0174    628 ; $EQU   TF$V_LBUF_SML   1               ; Local bfr too small SNAK set rsn 16
0174    629 ; $EQU   TF$M_LBUF_SML   2
0174    630 ;; $DEF   TF$B_LBE        .BLKB           ; Local buffer error
0174    631 ; $DEF   TF$W_RBETYP     .BLKW           ; NMA definition for the field
0174    632 ; $DEF   TF$W_RBEBC      .BLKW           ; Remote buffer error bit counters
0174    633 ; $EQU   TF$V_RBUF_NAVL  0               ; Remote buffer unavl NAK RCV'd rsn 8
0174    634 ; $EQU   TF$M_RBUF_NAVL  1
```

```
0174   635 ; $EQU  TF$V_RBUF_SML   1               ; Remote bfr too small NAK RCV'd rsn 16
0174   636 ; $EQU  TF$M_RBUF_SML   2
0174   637 ; $DEF  TF$B_RBE        .BLKB           ; Remote buffer error
0174   638 ; $DEF  TF$W_STOTYP     .BLKW           ; NMA definition for the fields
0174   639 ; $DEF  TF$W_STOBC      .BLKW           ; Selection timeout bit counters
0174   640 ; $EQU  TF$V_NOREP_SEL  0               ; When no attempt to respond was made
0174   641 ; $EQU  TF$M_NOREP_SEL  1
0174   642 ; $EQU  TF$V_INCREP_SEL 1               ; When attempt is made but the timeout
0174   643 ; $EQU  TF$M_INCREP_SEL 2
0174   644 ; $DEF  TF$B_STO        .BLKB           ; Selection timeout
0174   645 ; $DEF  TF$W_LRTOTYP    .BLKW           ; NMA definition for field
0174   646 ;; $DEF TF$B_LRTO       .BLKB           ; Records setting of SREP
0174   647 ; $DEF  TF$W_RRTOTYP    .BLKW           ; NMA definition for field
0174   648 ; $DEF  TF$B_RRTO       .BLKB           ; Records setting SACK when REP RCV'd
0174   649 ;                                       ; with NUMB = R
0174   650 ; $DEF  TF$W_THRES      .BLKW           ; Threshold errors
0174   651 ;                                       ; Bits 0-2 RCV threshold errors
0174   652 ;                                       ; Bits 3-5 XMT threshold errors
0174   653 ;                                       ; Bits 6-8 Selection threshold errors
0174   654 ;                       .BLKB   1       ; Reserved
0174   655 ;$DEF   GF$A_GFB                        ; Start of global control block
0174   656 ; $DEF  GF$B_STATE      .BLKB           ; State of protocol
0174   657 ; $EQU  GF$V_TIMER_RUNNING 0            ; 0 = Timer not running
0174   658 ; $EQU  GF$M_TIMER_RUNNING 1            ; 1 = timmer running
0174   659 ; $DEF  GF$B_TIMER_STATE .BLKB  1      ; State of DDCMP timer
0174   660 ; $EQU  GF$V_CRC         0             ; 0 = Device does CRC checking
0174   661 ; $EQU  GF$M_CRC         1
0174   662 ; $DEF  GF$B_DRVCHR     .BLKB           ; Device charateristics
0174   663 ; $DEF  GF$W_SELWAI     .BLKW           ; Amount of time to wait before timing
0174   664 ;                                       ; the selection interval out
0174   665 ; $DEF  GF$W_NEXTCNT    .BLKW   1       ; Count on which interval to deselect
0174   666 ;                                       ; the station when running in trib mode
0174   667 ; $DEF  GF$B_MAXSEL     .BLKB           ; Max number of selection intervals
0174   668 ; $DEF  GF$B_CURSEL     .BLKB           ; Current number of selection intervals
0174   669 ; $DEF  GF$B_FIPL       .BLKB   1       ; Fork IPL for DDCMP
0174   670 ; $DEF  GF$B_DIPL       .BLKB   1       ; Device IPL for DDCMP
0174   671 ; $DEF  GF$W_GEB        .BLKW   1       ; Size of global error block
0174   672 ; $DEF  GF$L_SELEND     .BLKL   1       ; Time the selection int expires
0174   673 ; $DEF  GF$L_TQE_STS    .BLKL   1       ; Timeout return status
0174   674 ; $DEF  GF$B_STRTIM     .BLKB           ; Amount of time to wait before setting
0174   675 ;                                       ; station streaming flag
0174   676 ; $DEF  GF$B_BABTIM     .BLKB           ; Amount of time to wait before setting
0174   677 ;                                       ; station babbling flag
0174   678 ; $DEF  GF$L_STRTMR     .BLKL           ; Stream timer
0174   679 ; $DEF  GF$L_BABTMR     .BLKL           ; Babbling timer
0174   680 ;
0174   681 ;; Station error counters
0174   682 ;
0174   683 ; $DEF  GF$K_ERRSRT                     ; Start of global error counters
0174   684 ;; $DEF GF$W_LSETYP     .BLKW           ; NMA definition for field
0174   685 ; $DEF  GF$W_LSE_BCTRS  .BLKW           ; Local station error bit counters
0174   686 ; $EQU  GF$V_LRCV_OVR   0               ; Receive overrun SNAK set reason 9
0174   687 ; $EQU  GF$M_LRCV_OVR   1
0174   688 ; $EQU  GF$V_LNRCV_OVR  1               ; Receive overrun SNAK not set
0174   689 ; $EQU  GF$M_LNRCV_OVR  2
0174   690 ; $EQU  GF$V_LXMT_UNDER 2               ; XMT underrun
0174   691 ; $EQU  GF$M_LXMT_UNDER 4
```

```
                     0174    692 ; $EQU  GF$V_LMSGHDR_FMT        3       ; NAK RCV'd reason code 17
                     0174    693 ; $EQU  GF$M_LMSGHDR_FMT        8
                     0174    694 ; $DEF  GF$B_LSE          .BLKB         ; Local station errors
                     0174    695 ;
                     0174    696 ; $DEF  GF$W_RSETYP       .BLKW         ; NMA definition for field
                     0174    697 ; $DEF  GF$W_RSE_BCTRS    .BLKW         ; Remote station error bit counters
                     0174    698 ; $EQU  GF$V_RRCV_OVR     0             ; NAK's RCV'd reason code 9
                     0174    699 ; $EQU  GF$M_RRCV_OVR     1
                     0174    700 ; $EQU  GF$V_RMHDR_FMT    1             ; SNAK set reason code 17
                     0174    701 ; $EQU  GF$M_RMHDR_FMT    2
                     0174    702 ; $EQU  GF$V_SADR_ERR     2             ; Message RCV'd by an unselected trib
                     0174    703 ; $EQU  GF$M_SADR_ERR     4
                     0174    704 ; $EQU  GF$V_STR_TRIB     3             ; Streaming trib
                     0174    705 ; $EQU  GF$M_STR_TRIB     8
                     0174    706 ; $DEF  GF$B_RSE          .BLKB         ; Remote station errors
                     0174    707 ; $DEF  GF$B_GH_CRC       .BLKB         ; Global header CRC error
                     0174    708 ; $DEF  GF$B_MDF_CRC      .BLKB         ; Maintenance data field CRC error
                     0174    709 ;                         .BLKB         ; spare
                     0174    710 ;
                     0174    711         .IF     DF ERR$$$
                     0174    712 $def    ucb$w_xg_xmtlat .blkw   1             ; count on XMT latency err's
                     0174    713 $def    ucb$w_xg_rcvlat .blkw   1             ; count on RCV latency err's
                     0174    714 $def    ucb$w_xg_xmtnxm .blkw   1             ; count on XMT nxm err's
                     0174    715 $def    ucb$w_xg_rcvnxm .blkw   1             ; count on RCV nxm err's
                     0174    716 $def    ucb$w_xg_strt   .blkw   1             ; count on START rcv'd errors
                     0174    717 $def    ucb$w_xg_prst   .blkw   1             ; count on perstitant errors
                     0174    718 $def    ucb$w_xg_disc   .blkw   1             ; count on modem disconnet err
                     0174    719                         .blkw   1             ; reserved
                     0174    720         .ENDC   ;DF ERR$$$
                     0174    721
                     0174    722         .IF     DF CTS$$$
                     0174    723         cts_size = 32                         ; set size of buffer to keep
                     0174    724                         .blkb   3             ; spare
                     0174    725 $def    ucb$b_cts_last  .blkb   1             ; set last used location
                     0174    726 $def    ucb$b_cts_buf   .blkb   cts_size      ; set size of buffer
                     0174    727         ucb$c_cts_len   =.-ucb$b_cts_last     ; set length
                     0174    728         .ENDC                                 ; DF CTS$$$ end
                     0174    729
                     0174    730         .IF     DF JNX$$$
                     0174    731                 .blkb   3
                     0174    732         jnx_size = 32
                     0174    733 $def    ucb$b_last      .blkb   1
                     0174    734 $def    ucb$l_buffer    .blkl   jnx_size
                     0174    735         ucb$c_last_len = .-ucb$b_last
                     0174    736         .ENDC   ;DF JNX$$$
                     0174    737
00000174             0174    738         UCB$C_XG_LENGTH =.              ; Size of XGDRIVER UCB
                     0174    739
                     0174    740
                     0174    741 ;
                     0174    742 ; Device status bits
                     0174    743 ;
                     0174    744
00000000             0174    745 .=0
                     0000    746         _VIELD  XG_DS,0,<-                    ; UCB$W_DEVSTS bits
                     0000    747                 <XMTING,,M>,-                 ; Device XMTing if 0
                     0000    748                 <RCVING,,M>,-                 ; Device RCVing if 0
```

```
                     0000   749                 <INITED,,M>,-                   ; Unit initialized
                     0000   750                 <FORK_PEND,,M>,-                ; Fork pending
                     0000   751                 <RCVPS,,M>,-                    ; On interrupt if bit is 0
                     0000   752                 <LOADRPS,,M>,-                  ; Which RCV buff to load next
                     0000   753                 <RCV_DONEP,,M>,-                ; Bit set when pri buff loaded
                     0000   754                 <RCV_DONES,,M>,-                ; Bit set when sec buff loaded
                     0000   755                 <XMTPS,,M>,-                    ; comp prim else comp secondary
                     0000   756                 <LOADXPS,,M>,-                  ; Which XMT buff to load next
                     0000   757                 <XMT_DONEP,,M>,-                ; Bit set when pri buff loaded
                     0000   758                 <XMT_DONES,,M>,-                ; Bit set when sec buff loaded
                     0000   759                 <RCVENB,,M>,-                   ; Set to enable the board
                     0000   760                 <XMTENB,,M>,-                   ; to rcv or xmt
                     0000   761                 <CTSTQE_RUN,,M>,-               ; Set when CTS TQE is queued
                     0000   762                 <CLEAN,,M>,-                    ; If XMT INPR when CLEAN req
                     0000   763                 >                               ; set saying abort after XMT'd
                     0000   764
                     0000   765 ;
                     0000   766 ; Receive buffer definition
                     0000   767 ;
                     0000   768 ; This structure must be the same size as the CXB
                     0000   769 ;
                     0000   770         $DEFINI RCV
00000000             0000   771 .=0
                     0000   772 $DEF     RCV_L_LINK      .BLKL   2               ; Forward and backward links
                     0008   773 $DEF     RCV_W_BLKSIZE   .BLKW   1               ; Total block size
                     000A   774 $DEF     RCV_B_BLKTYPE   .BLKB   1               ; Block type
                     000B   775 $DEF     RCV_B_FIPL      .BLKB   1               ; Fork IPL
                     000C   776 $DEF     RCV_W_MSGSIZ    .BLKW   1               ; Size of message received
                     000E   777 $DEF     RCV_W_ERROR     .BLKW   1               ; Error status
                     0010   778 $DEF     RCV_L_BACC      .BLKL   1               ; Buffer addr / char count
                     0014   779 $DEF     RCV_B_SLT       .BLKB   1               ; Receive slot number used
                     0015   780
                     0015   781 $DEF     RCV_CXB_SPARE   .BLKB   CXB$K_HEADER-XG$C_HEADER-.
                     0042   782                                                 ; Spare bytes to allow for CXB
                     0042   783 $DEF     RCV_Z_HEADER    .BLKB   XG$C_HEADER     ; Size in bytes of msg header
                     0048   784 $DEF     RCV_T_DATA                              ; Receive data
                     0048   785
                     0048   786         ASSUME  RCV_Z_HEADER+6 EQ CXB$K_HEADER
                     0048   787         ASSUME  RCV_T_DATA GE CXB$K_HEADER
                     0048   788
                     0048   789 ;
                     0048   790 ; P2 buffer header definition
                     0048   791 ;
                     0048   792         $DEFINI P2B
00000000             0000   793 .=0
                     0000   794 $DEF     P2B_L_POINTER   .BLKL   1       ; Pointer to start of data
                     0004   795 $DEF     P2B_L_BUFFER    .BLKL   1       ; Address of user's data buffer
                     0008   796 $DEF     P2B_W_SIZE      .BLKW   1       ; Size of P2 buffer
                     000A   797 $DEF     P2B_B_TYPE      .BLKB   1       ; Type of structure
                     000B   798 $DEF     P2B_B_SPARE     .BLKB   1       ; Spare byte
                     000C   799 $DEF     P2B_C_LENGTH                    ; Size of P2 buffer header
                     000C   800 $DEF     P2B_T_DATA                      ; Start of data
                     000C   801
                     000C   802         $DEFEND P2B
                     0048   803
                     0048   804
```

```
                     0048    806          .SBTTL  Standard Driver Tables
                     0048    807 ;
                     0048    808 ; Driver Prologue Table
                     0048    809 ;
                     0048    810          DPTAB   -                                       ;
                     0048    811                  END=DRIVER_END,-                        ; End of driver
                     0048    812                  ADAPTER=UBA,-                           ; UNIBUS device
                     0048    813                  UCBSIZE=UCB$C_XG_LENGTH,-               ; UCB size
                     0048    814                  NAME=XGDRIVER                           ; Driver name
                     0038    815
                     0038    816          DPT_STORE INIT                                  ; Initialization data
                     0038    817          DPT_STORE UCB,UCB$B_FIPL,B,8                    ; Fork IPL
                     003C    818          DPT_STORE UCB,UCB$B_DIPL,B,21                   ; Device IPL
                     0040    819          DPT_STORE UCB,UCB$L_DEVCHAR,L,<-                ; Device characteristics
                     0040    820                  DEV$M_AVL!DEV$M_NET!DEV$M_IDV!DEV$M_ODV>
                     0047    821          DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_SCOM         ; Device class
                     004B    822          DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_DMF32         ; Device type
                     004F    823          DPT_STORE UCB,UCB$W_DEVSTS,W,0                  ; Internal status
                     0054    824
                     0054    825          DPT_STORE REINIT                                ; Init data also for reload
                     0054    826          DPT_STORE DDB,DDB$L_DDT,D,XG$DDT                ; Driver dispatch table
                     0059    827          DPT_STORE CRB,CRB$L_INTD+4,D,RECEIVE_INTR       ; RCV intr service routine
                     005E    828          DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL,D,CONTROL_INIT ; Controller init
                     0063    829          DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,UNIT_INIT ; Unit init routine
                     0068    830          DPT_STORE CRB,CRB$L_INTD2+4,D,TRANSMIT_INTR     ; Transmit interrupt service
                     006D    831          DPT_STORE END
                     0048    832
                00000000     833          .PSECT  $$$115_DRIVER,LONG
                     0000    834
                     0000    835 ;
                     0000    836 ; Driver Dispatch Table
                     0000    837 ;
                     0000    838          DDTAB   DEVNAM=XG,-                             ; Device name
                     0000    839                  START=STARTIO,-                         ; Start I/O routine
                     0000    840                  FUNCTB=FUNCTABLE,-                      ; Function decision table
                     0000    841                  CANCEL=CANCEL,-                         ; Cancel I/O routine
                     0000    842                  REGDMP=REGDUMP,-                        ; Register dump routine
                     0000    843                  DIAGBF=<36+12>,-                        ; Diagnostic buffer size
                     0000    844                  ALTSTART=ALT_ENTRY                      ; Alternate entry routine
                     0038    845 ;
                     0038    846 ; Function Decision Table
                     0038    847 ;
                     0038    848 FUNCTABLE:
                     0038    849          FUNCTAB,-                                       ; Legal functions
                     0038    850                  <WRITEVBLK,WRITELBLK,WRITEPBLK,-        ; Transmit functions
                     0038    851                   READVBLK,READLBLK,READPBLK,-           ; Receive functions
                     0038    852                   SETMODE,SENSEMODE,SETCHAR,-            ; Set mode functions
                     0038    853                   CLEAN>                                 ; Clean for LAPB
                     0040    854          FUNCTAB,-                                       ; Buffered I/O functions
                     0040    855                  <WRITEVBLK,WRITELBLK,WRITEPBLK,-        ; Transmit functions
                     0040    856                   READLBLK,READPBLK,READVBLK,-           ;
                     0040    857                   SETMODE,SENSEMODE,SETCHAR,-            ; Set mode functions
                     0040    858                   CLEAN>                                 ; Clean for LAPB
                     0048    859
                     0048    860          .IF     DF JNX$$$
                     0048    861          functab fillbuffer,-                            ; Buffered I/O functions
                     0048    862                  <writevblk,writelblk,writepblk,-        ; Transmit functions
```

```
                0048    863                     readvblk,readlblk,readpblk,-    ; Receive functions
                0048    864                     setmode,sensemode,setchar,-     ; Set mode functions
                0048    865                     clean>                          ; Clean for LAPB
                0048    866             .ENDC   ;DF JNX$$$
                0048    867
                0048    868             FUNCTAB XMITFDT,-                       ; Transmit function dispatcher
                0048    869                     <WRITELBLK,WRITEPBLK,WRITEVBLK> ;
                0054    870             FUNCTAB RCVFDT,-                        ; Receive function dispatcher
                0054    871                     <READLBLK,READPBLK,READVBLK>    ;
                0060    872             FUNCTAB SETMODEFDT,-                    ; FDT for set mode and set char
                0060    873                     <SETMODE,SETCHAR>               ;
                006C    874             FUNCTAB SENSEMODEFDT,-                  ; FDT sensemode routine
                006C    875                     <SENSEMODE>
                0078    876             FUNCTAB CLEANFDT,-                      ; FDT clean routine
                0078    877                     <CLEAN>
                0084    878
```

```
                     0084    880          .SBTTL  P2 buffer verification tables
                     0084    881
                     0084    882 ;
                     0084    883 ; Define P2 buffer verification offsets
                     0084    884 ;
                     0084    885          $DEFINI PARAM
                     0000    886
                     0000    887          _VIELD  PRM,0,<-                        ; Parameter bits and sizes
                     0000    888                  <TYPE,12,M>,-                   ; Parameter type
                     0000    889                  <MIN,1,M>,-                     ; Parameter minimum value
                     0000    890                  <MAX,1,M>,-                     ; Parameter maximum value
                     0000    891                  <INVALID,1,M>,-                 ; Parameter invalid flags
                     0000    892                  >
                     0000    893
                     0000    894          _VIELD  OFF,0,<-                        ; Offset word fields
                     0000    895                  <VALUE,14,M>,-                  ; Offset value
                     0000    896                  <WIDTH,2,M>,-                   ; Size of field in structure
                     0000    897                  >
                     0000    898
                     0000    899          $DEFEND PARAM
                     0084    900
                     0084    901 ;
                     0084    902 ; Define UCB (line) parameters
                     0084    903 ;
00000000             0084    904 LINE_PRM_BUFSIZ = 0                             ; Line parameter buffer size
                     0084    905 LINE_PARAM:                                     ; Start of line parameters
                     0084    906
                     0084    907          PARAM   NMA$C_PCLI_PRO,-                ; Protocol selection
                     0084    908                  OFFSET=UCB$B_XG_PRO,WIDTH=B,-
                     0084    909                  MAX=NMA$C_LINPR_BSY,-
                     0084    910                  INVALID=XG_DS_M_INITED,-        ; Device can't be ON
                     0084    911                  BASE=LINE
                     008C    912
                     008C    913          PARAM   NMA$C_PCLI_DUP,-                ; Duplex mode
                     008C    914                  OFFSET=UCB$B_XG_DUP,WIDTH=B,-
                     008C    915                  MAX=NMA$C_DPX_HAL,-
                     008C    916                  INVALID=XG_DS_M_INITED,-        ; Device can't be ON
                     008C    917                  BASE=LINE
                     0094    918
                     0094    919          PARAM   NMA$C_PCLI_CON,-                ; Controller mode
                     0094    920                  OFFSET=UCB$B_XG_CON,WIDTH=B,-
                     0094    921                  MAX=NMA$C_LINCN_LOO,-
                     0094    922                  INVALID=XG_DS_M_INITED,-        ; Device can't be On
                     0094    923                  BASE=LINE
                     009C    924
                     009C    925          PARAM   NMA$C_PCLI_BFN,-                ; Number of receives
                     009C    926                  OFFSET=UCB$B_XG_BFN,WIDTH=B,-
                     009C    927                  MIN=1,MAX=255,-
                     009C    928                  INVALID=XG_DS_M_INITED,-        ; Device can't be ON
                     009C    929                  BASE=LINE
                     00A6    930
                     00A6    931          PARAM   NMA$C_PCLI_BUS,-                ; Buffer size
                     00A6    932                  OFFSET=UCB$W_DEVBUFSIZ,WIDTH=W,-
                     00A6    933                  MIN=1,MAX=16383,-
                     00A6    934                  INVALID=XG_DS_M_INITED,-        ; Device can't be ON
                     00A6    935                  BASE=LINE
                     00B0    936
```

```
00B0   937          PARAM   NMA$C_PCLI_RTT,-                    ; Retransmit timer
00B0   938                  OFFSET=UCB$Z_XG_DDCMP+DLK$W_REPWAIT,-
00B0   939                  WIDTH=W,-
00B0   940                  MIN=50,-
00B0   941                  BASE=LINE
00B6   942
00B6   943          PARAM   NMA$C_PCLI_FRA,-                    ; Address of BISYNC
00B6   944                  OFFSET=UCB$A_XG_FRAME_ADDR,-        ; framing routine
00B6   945                  WIDTH=L,-
00B6   946                  INVALID=XG_DS_M_INITED,-            ; Device can't be ON
00B6   947                  BASE=LINE
00BC   948
00BC   949          PARAM   NMA$C_PCLI_STI1,-                   ; BISYNC state info 1ST
00BC   950                  OFFSET=UCB$L_XG_STATE_INFO1,-       ; longword
00BC   951                  WIDTH=L,-
00BC   952                  INVALID=XG_DS_M_INITED,-            ; Device can't be ON
00BC   953                  BASE=LINE
00C2   954
00C2   955          PARAM   NMA$C_PCLI_STI2,-                   ; BISYNC state info 2ND
00C2   956                  OFFSET=UCB$L_XG_STATE_INFO2,-       ; longword
00C2   957                  WIDTH=L,-
00C2   958                  INVALID=XG_DS_M_INITED,-            ; Device can't be ON
00C2   959                  BASE=LINE
00C8   960
00C8   961          PARAM   NMA$C_PCLI_TMO,-                    ; Timeout value
00C8   962                  OFFSET=UCB$B_XG_TIMEOUT,-
00C8   963                  WIDTH=B,-
00C8   964                  INVALID=XG_DS_M_INITED,-            ; Device can't be ON
00C8   965                  BASE=LINE
00CE   966
00CE   967          PARAM   NMA$C_PCLI_MCL,-                    ; Drop DTR on deassign
00CE   968                  OFFSET=UCB$B_XG_MODEM_CLR,-         ; except last deassign, where
00CE   969                  WIDTH=B,-                           ; DTR is always droped
00CE   970                  MIN=0,-
00CE   971                  MAX=1,-
00CE   972                  INVALID=XG_DS_M_INITED,-            ; Device can't be ON
00CE   973                  BASE=LINE
00D8   974
00D8   975  ; Define DMF32 specific parameters
00D8   976
00D8   977          PARAM   NMA$C_PCLI_SYC,-                    ; Value for the sync char
00D8   978                  OFFSET=UCB$Z_XG_SYNC+XG$B_SYNC_REG,-
00D8   979                  WIDTH=B,-
00D8   980                  INVALID=XG_DS_M_INITED,-
00D8   981                  BASE=LINE
00DE   982
00DE   983          PARAM   NMA$C_PCLI_NMS,-                    ; Number of sync char to send
00DE   984                  OFFSET=UCB$Z_XG_SYNC+XG$B_NUM_SYNC,-
00DE   985                  WIDTH=B,-
00DE   986                  INVALID=XG_DS_M_INITED,-
00DE   987                  BASE=LINE
00E4   988
00E4   989          PARAM   NMA$C_PCLI_BPC,-                    ; Number of bits per char
00E4   990                  OFFSET=UCB$Z_XG_SYNC+XG$B_BPC,-
00E4   991                  WIDTH=B,-
00E4   992                  MIN=5,-
00E4   993                  MAX=8,-
```

```
                        00E4   994                  INVALID=XG_DS_M_INITED,-
                        00E4   995                  BASE=LINE
                        00EE   996
                        00EE   997          PARAM   NMA$C_PCLI_MNTL,-                   ; Maintenance loopback mode
                        00EE   998                  OFFSET=UCB$Z_XG_SYNC+XG$B_MNTLOOP,- ; for devices which support
                        00EE   999                  WIDTH=B,-                           ; several different loopback
                        00EE  1000                  MAX=NMA$C_PCLI_INTL3,-              ; modes
                        00EE  1001                  INVALID=XG_DS_M_INITED,-
                        00EE  1002                  BASE=LINE
                        00F6  1003
                        00F6  1004          PARAM                                       ; End of table
                        00F8  1005 ;
                        00F8  1006 ; Define Trib parameters
                        00F8  1007 ;
                00000000 00F8  1008 TRIB_PRM_BUFSIZ  = 0                                ; Trib parameter buffer size
                        00F8  1009 TRIB_PARAM:                                          ; Start of trib parameters
                        00F8  1010
                        00F8  1011          PARAM   NMA$C_PCCI_TRI,-                    ; Trib address
                        00F8  1012                  OFFSET=DLK$B_TRIB,WIDTH=B,-
                        00F8  1013                  MIN=1,-
                        00F8  1014                  INVALID=XM$M_STS_ACTIVE,-           ; Trib can't be established
                        00F8  1015                  BASE=TRIB
                        0100  1016
                        0100  1017          PARAM   NMA$C_PCCI_MTR,-                    ; Max number of messages sent
                        0100  1018                  OFFSET=DLK$B_MSGCNT,WIDTH=B,-       ; in a selection interval
                        0100  1019                  MIN=1,-
                        0100  1020                  MAX=100,-
                        0100  1021                  BASE=TRIB
                        0108  1022
                        0108  1023          PARAM   NMA$C_PCCI_MST,-                    ; Maintenance state
                        0108  1024                  OFFSET=DLK$B_MAINT,WIDTH=B,-
                        0108  1025                  MAX=NMA$C_STATE_OFF,-
                        0108  1026                  BASE=TRIB
                        010E  1027
                        010E  1028          PARAM   NMA$C_PCCI_MRB,-                    ; Maxium receive buffers
                        010E  1029                  OFFSET=DLK$B_MRB,WIDTH=B,-
                        010E  1030                  MIN=1,-
                        010E  1031                  INVALID=XM$M_STS_ACTIVE,-
                        010E  1032                  BASE=TRIB
                        0116  1033
                        0116  1034          PARAM                                       ; End of trib tables
                        0118  1035
```

```
                     0118  1037
                     0118  1038 ;
                     0118  1039 ; DEFAULT TRIBUTARY PARAMETERS for DDCMP
                     0118  1040 ;
                     0118  1041 DEF_TRIB_PARAM::
                     0118  1042         ASSUME  DLK$B_MSGCNT EQ DLK$B_TRIB+1
                     0118  1043         ASSUME  DLK$B_MAXREP EQ DLK$B_MSGCNT+1
                     0118  1044         ASSUME  DLK$B_MAXSEL EQ DLK$B_MAXREP+1
                     0118  1045         ASSUME  DLK$W_REPWAIT EQ DLK$B_MAXSEL+1
                     0118  1046         ASSUME  DLK$W_SELWAIT EQ DLK$W_REPWAIT+2
                     0118  1047         ASSUME  DLK$B_MAINT EQ DLK$W_SELWAIT+2
                     0118  1048         ASSUME  DLK$B_MRB EQ DLK$B_MAINT+1
                  01 0118  1049         .BYTE   1                               ; Default Trib address
                  04 0119  1050         .BYTE   4                               ; Max nmb of msgs sent / select
                  04 011A  1051         .BYTE   4                               ; Max nmb of sel intls allowed
                  03 011B  1052         .BYTE   3                               ; Max times to reXMT a msg
                0BB8 011C  1053         .WORD   3000                            ; Reply timeout timer
                0BB8 011E  1054         .WORD   3000                            ; Selection timer in sec
                  01 0120  1055         .BYTE   NMA$C_STATE_OFF                 ; Default is no maint mode
                  FF 0121  1056         .BYTE   255                             ; Default to unlimited rcv buff
          0000000A 0122  1057         DEF_TRIB_PARAMSZ = .-DEF_TRIB_PARAM
                     0122  1058 ;
                     0122  1059 ; Default line parameter values
                     0122  1060 ;
                     0122  1061 DEF_LINE_PARAM::
                     0122  1062         ASSUME  UCB$B_XG_PRO EQ UCB$B_XG_SETPRM
                     0122  1063         ASSUME  UCB$B_XG_DUP EQ UCB$B_XG_PRO+1
                     0122  1064         ASSUME  UCB$B_XG_CON EQ UCB$B_XG_DUP+1
                     0122  1065         ASSUME  UCB$B_XG_BFN EQ UCB$B_XG_CON+1
                     0122  1066         ASSUME  UCB$B_XG_SPD EQ UCB$B_XG_BFN+1
                     0122  1067         ASSUME  UCB$B_XG_TIMEOUT EQ UCB$B_XG_SPD+1
                     0122  1068         ASSUME  UCB$B_XG_MODEM_CLR EQ UCB$B_XG_TIMEOUT+1
                     0122  1069         ASSUME  UCB$B_XG_MNT_LOOPB EQ UCB$B_XG_MODEM_CLR+1
                     0122  1070         ASSUME  UCB$A_XG_FRAME_ADDR EQ UCB$B_XG_MNT_LOOPB+1
                     0122  1071         ASSUME  UCB$Q_XG_STATE_INFO EQ UCB$A_XG_FRAME_ADDR+4
                  00 0122  1072         .BYTE   NMA$C_LINPR_POI                 ; Protocol is point-point
                  00 0123  1073         .BYTE   NMA$C_DPX_FUL                   ; Duplex is full
                  00 0124  1074         .BYTE   NMA$C_LINCN_NCR                 ; Controller mode is normal
                  01 0125  1075         .BYTE   1                               ; Number of receive buffers
                     0126  1076 ;& Currently not implemented
                  00 0126  1077         .BYTE   0                               ; Set default line speed
                  06 0127  1078         .BYTE   XG$C_WFCTS_SEC                  ; Timeout on CTS
                  00 0128  1079         .BYTE   0                               ; Modem status after deassign
                  00 0129  1080         .BYTE   0                               ; Type of loopb
          00000000 012A  1081         .LONG   0                               ; Address of BISYNC framing routine
00000000  00000000 012E  1082         .QUAD   0                               ; BISYNC state info
                     0136  1083
                     0136  1084
          00000014   0136  1085         DEF_LINE_PARAMSZ = .-DEF_LINE_PARAM
```

```
                        0136  1087 ;
                        0136  1088 ; Default device parameters for DDCMP
                        0136  1089 ;
                        0136  1090 DEF_SYNC_PARAM::
                        0136  1091          ASSUME  XG$B_PROTOCOL EQ XG$B_ERR_CNTRL+1
                        0136  1092          ASSUME  XG$B_TX_BPC EQ XG$B_PROTOCOL+1
                        0136  1093          ASSUME  XG$B_RX_BPC EQ XG$B_TX_BPC+1
                        0136  1094          ASSUME  XG$B_BAUD EQ XG$B_RX_BPC+1
                        0136  1095          ASSUME  XG$B_NUM_SYNC EQ XG$B_BAUD+1
                        0136  1096          ASSUME  XG$B_SYNC_REG EQ XG$B_NUM_SYNC+1
                        0136  1097          ASSUME  XG$B_ICLK EQ XG$B_SYNC_REG+1
                  03    0136  1098          .BYTE   XG$C_ERR_CRC16                  ; Error control CRC16
                  00    0137  1099          .BYTE   XG$C_PRO_DDCMP                  ; Protocol is DDCMP
                  00    0138  1100          .BYTE   XG$C_BPC_8                      ; Bits per char - 8
                  00    0139  1101          .BYTE   XG$C_BPC_8                      ; Bits per char - 8
                  07    013A  1102          .BYTE   XG$C_BRG_19200                  ; Line speed 19.2 kbs
                  08    013B  1103          .BYTE   8                               ; Number of syncs to send
                  96    013C  1104          .BYTE   XG$C_SYNC_DDCMP                 ; Sync character
                  00    013D  1105          .BYTE   XG$C_INTCLK_OFF                 ; No internal clock
           00000008     013E  1106          DEF_SYNC_PARAMSZ = .-DEF_SYNC_PARAM
                        013E  1107
                        013E  1108 ;
                        013E  1109 ; Default device params for LAPB
                        013E  1110 ;
                        013E  1111 DEF_LAPB_PARAM::
                  00    013E  1112          .BYTE   XG$C_ERR_CRC1                   ; Err control CRC-CCITT to 1's
                  02    013F  1113          .BYTE   XG$C_PRO_HDLC                   ; Set protocol
                  00    0140  1114          .BYTE   XG$C_BPC_8                      ; Set BPC rcv
                  00    0141  1115          .BYTE   XG$C_BPC_8                      ; Set BPC xmt
                  07    0142  1116          .BYTE   XG$C_BRG_19200                  ; Line speed 19.2 kbs
                  00    0143  1117          .BYTE   0                               ; No syncs to send
                  00    0144  1118          .BYTE   XG$C_SYNC_HDLC                  ; no sync character
                  00    0145  1119          .BYTE   XG$C_INTCLK_OFF                 ; Set internal clock off
           00000008     0146  1120          DEF_LAPB_PARAMSZ = .-DEF_LAPB_PARAM
                        0146  1121 ;
                        0146  1122 ; Default device parameters for BISYNC
                        0146  1123 ;
                        0146  1124 DEF_BISYNC_PARAM::
                  07    0146  1125          .BYTE   XG$C_NOCON                      ; No error control
                  07    0147  1126          .BYTE   XG$C_GENBYTE                    ; Set protocol type
                  00    0148  1127          .BYTE   XG$C_BPC_8                      ; Set default rcv bits per char
                  00    0149  1128          .BYTE   XG$C_BPC_8                      ; Set default xmt bits per char
                  07    014A  1129          .BYTE   XG$C_BRG_19200                  ; Set default line speed
                  08    014B  1130          .BYTE   8                               ; Set default number of syncs
                  32    014C  1131          .BYTE   XG$C_SYNC_BISYNC                ; Set sync char
                  00    014D  1132          .BYTE   XG$C_INTCLK_OFF                 ; Set no internal clock
           00000008     014E  1133          DEF_BISYNC_PARAMSZ =.-DEF_BISYNC_PARAM
                        014E  1134
```

```
                                014E  1136          .SBTTL  CONTROL_INIT - Initialize Sync line device
                                014E  1137 ;++
                                014E  1138 ; CONTROL_INIT - Initialize Sync line unit
                                014E  1139 ;
                                014E  1140 ; FUNCTINAL DESCRIPTION:
                                014E  1141 ;
                                014E  1142 ; This routine is entered at SYSTARTUP to initialize the sync line. The action
                                014E  1143 ; is to set the interrupt vector on the combo.
                                014E  1144 ;
                                014E  1145 ; INPUTS:
                                014E  1146 ;       R4 = Address of device CSR
                                014E  1147 ;       R5 = Address of device IDB
                                014E  1148 ;       R6 = Address of device DDB
                                014E  1149 ;       R8 = Address of device CRB
                                014E  1150 ;
                                014E  1151 ; OUTPUTS:
                                014E  1152 ;       R4,R5,R8 are preserved
                                014E  1153 ;--
                                014E  1154 CONTROL_INIT:
                                014E  1155
                                014E  1156 ; Set interrupt vector in controller
                                014E  1157
      50    0F A5    98         014E  1158          CVTBL   IDB$B_COMBO_CSR_OFFSET(R5),R0
            10 A5    83         0152  1159          SUBB3   IDB$B_COMBO_VECTOR_OFFSET(R5),-
    6440    0B A5                0155  1160                  IDB$B_VECTOR(R5),(R4)[R0]
                     05         0159  1161          RSB
                                015A  1162
```

```
                                015A  1164          .SBTTL  UNIT_INIT - Initialize the device unit
                                015A  1165 ;++
                                015A  1166 ; UNIT_INIT - Initialize the device unit
                                015A  1167 ;
                                015A  1168 ; FUNCTIONAL DESCRIPTION:
                                015A  1169 ;
                                015A  1170 ; This routine is called when the driver is loaded and during powerfailure
                                015A  1171 ; recovery.  It sets the unit status to ONLINE.  Also, if called during
                                015A  1172 ; powerfail recovery, it shuts down the device.
                                015A  1173 ;
                                015A  1174 ; INPUTS:
                                015A  1175 ;
                                015A  1176 ;       R4 = Address of the device CSR
                                015A  1177 ;       R5 = UCB address
                                015A  1178 ;
                                015A  1179 ; OUTPUTS:
                                015A  1180 ;
                                015A  1181 ;       R5 preserved
                                015A  1182 ;--
                                015A  1183 UNIT_INIT:                                           ; Initialize the unit
                                015A  1184          .IF     DF JNX$$$
                                015A  1185          pushr   #^m<r0,r1,r2,r3,r4,r5>
                                015A  1186          movc5   #0,ucb$l_buffer(r5),#0,#ucb$c_last_len,ucb$l_buffer(r5)
                                015A  1187          popr    #^m<r0,r1,r2,r3,r4,r5>
                                015A  1188          .ENDC   ;DF JNX$$$
                                015A  1189
                                015A  1190          .IF     DF DBG$$$
                                015A  1191          jsb     g^ini$brk
                                015A  1192          .ENDC   ;DF DBG$$$
                                015A  1193
            16 64 A5    05  E1  015A  1194          BBC     #UCB$V_POWER,UCB$W_STS(R5),10$  ; Br if not powerfail recovery
                        0B  E1  015F  1195          BBC     #XM$V_STS_ACTIVE,-              ; Br if not previously active
                 11 44 A5        0161  1196                  UCB$L_DEVDEPEND(R5),10$         ;
                        0F  BB  0164  1197          PUSHR   #^M<R0,R1,R2,R3>                ; Save all registers
             54    0364 8F  3C  0166  1198          MOVZWL  #SS$_POWERFAIL,R4               ; Indicate powerfail
                    189C  30  016B  1199          BSBW    SCHED_FORK
                      0F  BA  016E  1200          POPR    #^M<R0,R1,R2,R3>
                      05  0170  1201          RSB
                          0171  1202
                          0171  1203 5$:       BUG_CHECK NOBUFPCKT,FATAL
                          0175  1204
                          0175  1205 ;
                          0175  1206 ; This driver makes the assumption that IPL$_SYNC, IPL$_TIMER and the drivers
                          0175  1207 ; fork IPL are all equal. If any of these change the integrity of the driver
                          0175  1208 ; can not be assured. THIS DRIVER WILL NOT WORK ON PRE VERSION 4 SYSTEMS.
                          0175  1209 ;
                          0175  1210
                          0175  1211          ASSUME IPL$_SYNCH EQ IPL$_TIMER
                          0175  1212
                          0175  1213 ;
                          0175  1214 ; Also ASSUME that the following fields are equal, else the driver will get
                          0175  1215 ; very confused as to where transmits and completes are coming from when
                          0175  1216 ; running in any protocol mode other than DDCMP.
                          0175  1217
                          0175  1218          ASSUME  LAPB$Q_XMTQ EQ TF$Q_CTLQ
                          0175  1219          ASSUME  LAPB$Q_CLEANQ EQ TF$Q_CMPQ
                          0175  1220          ASSUME  LAPB$Q_BLANK EQ TF$Q_XMTQ
```

```
                                0175  1221          ASSUME  LAPB$B_XQCNT EQ TF$B_XQCNT
                                0175  1222          ASSUME  LAPB$Q_CLEANQ EQ BISYNC$Q_CLEANQ
                                0175  1223          ASSUME  BISYNC$Q_BLANK EQ TF$Q_CTLQ
                                0175  1224          ASSUME  BISYNC$Q_CLEANQ EQ TF$Q_CMPQ
                                0175  1225          ASSUME  BISYNC$Q_XMTQ EQ TF$Q_XMTQ
                                0175  1226          ASSUME  BISYNC$B_XQCNT EQ TF$B_XQCNT
                                0175  1227 ;
                                0175  1228 ; We must also be sure that SYNCH and the drivers FIPL are the same. If this
                                0175  1229 ; changes the integrity of the driver can not be assured with out some
                                0175  1230 ; major changes.
                                0175  1231 ;
                   0B A5   08   91   0175  1232 10$:     CMPB    #IPL$_SYNCH,UCB$B_FIPL(R5)      ; If the IPLs are not equal then
                           F6   12   0179  1233          BNEQ    5$                              ; cause a fatal bugcheck
                      0FF0 8F   BB   017B  1234          PUSHR   #^M<R4,R5,R6,R7,R8,R9,R10,R11>
                                017F  1235
                                017F  1236 ;
                                017F  1237 ; Set queue headers
                                017F  1238 ;
            00DC C5     00DC C5   9E   017F  1239          MOVAB   UCB$Q_XG_RCVS(R5),UCB$Q_XG_RCVS(R5)     ; Receive list
            00E0 C5     00DC C5   9E   0186  1240          MOVAB   UCB$Q_XG_RCVS(R5),UCB$Q_XG_RCVS+4(R5)
            00E4 C5     00E4 C5   9E   018D  1241          MOVAB   UCB$Q_XG_FREE(R5),UCB$Q_XG_FREE(R5)     ; Free buffer list
            00E8 C5     00E4 C5   9E   0194  1242          MOVAB   UCB$Q_XG_FREE(R5),UCB$Q_XG_FREE+4(R5)
            00EC C5     00EC C5   9E   019B  1243          MOVAB   UCB$Q_XG_RCV_INPR(R5),UCB$Q_XG_RCV_INPR(R5) ; Rcv inpr list
            00F0 C5     00EC C5   9E   01A2  1244          MOVAB   UCB$Q_XG_RCV_INPR(R5),UCB$Q_XG_RCV_INPR+4(R5)
            00F4 C5     00F4 C5   9E   01A9  1245          MOVAB   UCB$Q_XG_POST(R5),UCB$Q_XG_POST(R5)     ; Post list
            00F8 C5     00F4 C5   9E   01B0  1246          MOVAB   UCB$Q_XG_POST(R5),UCB$Q_XG_POST+4(R5)
            00D4 C5     00D4 C5   9E   01B7  1247          MOVAB   UCB$Q_XG_ATTN(R5),UCB$Q_XG_ATTN(R5)     ; Full buffer list
            00D8 C5     00D4 C5   9E   01BE  1248          MOVAB   UCB$Q_XG_ATTN(R5),UCB$Q_XG_ATTN+4(R5)
                        00FC C5   7C   01C5  1249          CLRQ    UCB$Z_XG_XMT_INPR(R5)
                        0090 C5   D4   01C9  1250          CLRL    UCB$L_XG_AST(R5)                        ; Attention AST listhead
                                  01CD  1251
                                  01CD  1252          .IF     DF ERR$$$
                                  01CD  1253          ASSUME  UCB$W_XG_RCVLAT EQ UCB$W_XG_XMTLAT+2
                                  01CD  1254          CLRL    UCB$W_XG_XMTLAT(R5)             ; Set no errors on device
                                  01CD  1255          .ENDC   ;DF ERR$$$
                                  01CD  1256
                             1A   10   01CD  1257          BSBB    INIT_PARAM                      ; Set all default values in device
                     54      0094 C5   DE   01CF  1258          MOVAL   UCB$L_XG_TQE(R5),R4             ; Set the CTS TQE address
20 A4       00000000 000186A0 8F   7D   01D4  1259          MOVQ    #XG$C_CTS_DELTA,TQE$Q_DELTA(R4) ; Set the delta time
                   64 A5     10   A8   01E0  1260          BISW    #UCB$M_ONLINE,UCB$W_STS(R5)     ; Set software status ONLINE
                      0FF0 8F   BA   01E4  1261          POPR    #^M<R4,R5,R6,R7,R8,R9,R10,R11>
                                05   01E8  1262          RSB
                                  01E9  1263
```

```
                                01E9  1265          .SBTTL  INIT_PARAM,  Initialize parameters
                                01E9  1266
                                01E9  1267 ;++
                                01E9  1268 ; INIT_PARAM - Initialize parameters
                                01E9  1269 ;
                                01E9  1270 ; Functional description:
                                01E9  1271 ;
                                01E9  1272 ; This routine is called to reset parameters when the unit is initialized.
                                01E9  1273 ;
                                01E9  1274 ; Inputs:
                                01E9  1275 ;
                                01E9  1276 ;       R5 = UCB address
                                01E9  1277 ;
                                01E9  1278 ;       IRP = FIPL or higher
                                01E9  1279 ;
                                01E9  1280 ; Outputs:
                                01E9  1281 ;
                                01E9  1282 ;       R0-R2 are destroyed.
                                01E9  1283 ;
                                01E9  1284 ;--
                                01E9  1285
                                01E9  1286 INIT_PARAM:                                  ; Initialize the UCB
        0100 8F     B0          01E9  1287          MOVW    #XG_DEF_BUFSIZ,-            ; Set default buffer size
          42 A5                 01ED  1288                  UCB$W_DEVBUFSIZ(R5)
        50    14    3C          01EF  1289          MOVZWL  #DEF_LINE_PARAMSZ,R0        ; Set size of defaults in bytes
51      FF2C CF     9E          01F2  1290          MOVAB   DEF_LINE_PARAM,R1           ; Set address of defaults
52      0150 C5     9E          01F7  1291          MOVAB   UCB$B_XG_SETPRM(R5),R2      ; Set address of parameters
        82    81    90          01FC  1292 10$:     MOVB    (R1)+,(R2)+                 ; Set next default
          FA 50     F5          01FF  1293          SOBGTR  R0,10$                      ; Loop on all parameters
        50    0A    3C          0202  1294          MOVZWL  #DEF_TRIB_PARAMSZ,R0        ; Set size of defaults in bytes
51      FF0F CF     9E          0205  1295          MOVAB   DEF_TRIB_PARAM,R1           ; Set address of defaults
52      0134 C5     9E          020A  1296          MOVAB   UCB$Z_XG_DDCMP(R5),R2       ; Set address of parameters
        82    81    90          020F  1297 20$:     MOVB    (R1)+,(R2)+                 ; Set next default
          FA 50     F5          0212  1298          SOBGTR  R0,20$                      ; Loop on all parameters
        50    08    3C          0215  1299          MOVZWL  #DEF_SYNC_PARAMSZ,R0        ; Set size of defaults in bytes
51      FF1A CF     9E          0218  1300          MOVAB   DEF_SYNC_PARAM,R1           ; Set address of defaults
52      0144 C5     9E          021D  1301          MOVAB   UCB$Z_XG_SYNC(R5),R2        ; Set address of parameters
        82    81    90          0222  1302 30$:     MOVB    (R1)+,(R2)+                 ; Set next default
          FA 50     F5          0225  1303          SOBGTR  R0,30$                      ; Loop on all parameters
                    05          0228  1304          RSB
```

```
                                    K 4
0229  1306          .IF     DF JNX$$$
0229  1307 fillbuffer:
0229  1308          movzbl  ucb$b_last(r5),r0
0229  1309          movl    r3,ucb$l_buffer(r5)[r0]
0229  1310          movq    #0,irp$l_media(r3)
0229  1311          incl    r0
0229  1312          bicl    #^c<jnx_size-1>,r0
0229  1313          movb    r0,ucb$b_last(r5)
0229  1314          rsb
0229  1315          .ENDC   ;DF JNX$$$
0229  1316
```

```
                     0229  1318          .SBTTL XMITFDT - Transmit I/O FDT routine
                     0229  1319 ;++
                     0229  1320 ; XMITFDT - Transmit I/O FDT routine
                     0229  1321 ;
                     0229  1322 ; FUNCTIONAL DESCRIPTION:
                     0229  1323 ;
                     0229  1324 ; This routine allocates a system buffer for the QIO. Then calls the
                     0229  1325 ; common FDT routine give the buffer to DDCMP.
                     0229  1326 ;
                     0229  1327 ; INPUTS.
                     0229  1328 ;
                     0229  1329 ;       R3 = I/O packet address
                     0229  1330 ;       R4 = Current PCB address
                     0229  1331 ;       R5 = UCB address
                     0229  1332 ;       R6 = CCB address
                     0229  1333 ;
                     0229  1334 ; OUTPUTS:
                     0229  1335 ;
                     0229  1336 ;       R3,R4,R5,R6,R7,R8,R9 are preserved
                     0229  1337 ;
                     0229  1338 ;--
                     0229  1339 XMITFDT:                                        ; Transmit FDT routine
     03F8 8F    BB   0229  1340          PUSHR   #^M<R3,R4,R5,R6,R7,R8,R9>
0167 C5    01    91   022D  1341          CMPB    #XG$C_PROTYPE_BISYNC,UCB$B_XG_PROTYPE(R5) ; BISYNC?
           21    12   0232  1342          BNEQ    4$                              ; Branch if not
  50   0C AC    D0   0234  1343          MOVL    P4(AP),R0                       ; See if any state info was specifed
           1B    13   0238  1344          BEQL    4$                              ; IF EQL then none
     51    08    D0   023A  1345          MOVL    #8,R1                           ; Set size of buffer to check
 00000000'GF    16   023D  1346          JSB     G^EXE$WRITECHK                  ; Check for write access
     50    60    7D   0243  1347          MOVQ    (R0),R0                         ; Mov the info into R0 and R1
                     0246  1348          DSBINT  UCB$B_FIPL(R5)                  ; SYnc to gain access to UCB
015C C5    50    7D   024D  1349          MOVQ    R0,UCB$Q_XG_STATE_INFO(R5)      ; Move the info into the UCB
                     0252  1350          ENBINT                                  ; Resync to lower IPL
     57    6C    D0   0255  1351 4$:      MOVL    P1(AP),R7                       ; Get address of buffer
  59   04 AC    D0   0258  1352          MOVL    P2(AP),R9                       ; Get the size of buffer
           68    13   025C  1353          BEQL    10$                             ; If EQL then size is zero
  42 A5    59    B1   025E  1354          CMPW    R9,UCB$W_DEVBUFSIZ(R5)          ; If GTR then report as error
           62    1A   0262  1355          BGTRU   10$
     50    57    D0   0264  1356          MOVL    R7,R0                           ; Set up R0 and R1 for
     51    59    D0   0267  1357          MOVL    R9,R1                           ; write access check
 00000000'GF    16   026A  1358          JSB     G^EXE$WRITECHK                  ; Check the acc of users buffer
51   59    2A   C1   0270  1359          ADDL3   #XMTQ$K_LENGTH,R9,R1            ; Get length of buffer to alloc
     56    53   D0   0274  1360          MOVL    R3,R6                           ; Save the IRP address
 00000000'GF    16   0277  1361          JSB     G^EXE$BUFFRQUOTA                ; Check quota
        49 50    E9   027D  1362          BLBC    R0,20$                          ; If LBC not enough quota
 00000000'GF    16   0280  1363          JSB     G^EXE$ALLOCBUF                  ; Allocated the buffer
        40 50    E9   0286  1364          BLBC    R0,20$                          ; If LBC not buffer allocated
     53    56    D0   0289  1365          MOVL    R6,R3                           ; Retreive the IRP
     58    52    D0   028C  1366          MOVL    R2,R8                           ; Get buffer address in R8
50   0080 C4    D0   028F  1367          MOVL    PCB$L_JIB(R4),R0                ; Get the JIB address
  20 A0    51    C2   0294  1368          SUBL2   R1,JIB$L_BYTCNT(R0)             ; Adjust buffered I/O
  30 A3    51    B0   0298  1369          MOVW    R1,IRP$W_BOFF(R3)               ; Save byte offset
  2C A3    52    D0   029C  1370          MOVL    R2,IRP$L_SVAPTE(R3)             ; Save system PTE
        1F A8    94   02A0  1371          CLRB    XMTQ$B_FLAG(R8)                 ; Clr bits, not "Internal" IRP
50   0167 C5    9A   02A3  1372          MOVZBL  UCB$B_XG_PROTYPE(R5),R0         ; Get protocol type
                     02A8  1373          CASE    R0,TYPE=B,<-                    ; Case on protocol type
                     02A8  1374                  30$,-                           ; LAPB
```

```
                        02A8  1375                      40$,-                                   ; BISYNC
                        02A8  1376                      >                                       ; Fall thru on DDCMP
            008F    30  02B0  1377 5$:          BSBW    COP_BUFF                                ; Branch to copy into my buff
                        02B3  1378              SETIPL  UCB$B_FIPL(R5)                          ; Sync to FIPL
              34    10  02B7  1379              BSBB    COM_XMITFDT                             ; Branch to common processing
           0D 50    E9  02B9  1380              BLBC    R0,20$                                  ; If BC then error
        03F8 8F     BA  02BC  1381 8$:          POPR    #^M<R3,R4,R5,R6,R7,R8,R9>
00000000'GF         17  02C0  1382              JMP     G^EXE$QIORETURN
                        02C6  1383
        50   14     3C  02C6  1384 10$:         MOVZWL  S^#SS$_BADPARAM,R0                      ; Set status for abort
        03F8 8F     BA  02C9  1385 20$:         POPR    #^M<R3,R4,R5,R6,R7,R8,R9>
00000000'GF         17  02CD  1386              JMP     G^EXE$ABORTIO
                        02D3  1387
            015C    30  02D3  1388 30$:         BSBW    COM_XMITLAPB                            ; Call LAPB FDT routine
        03F8 8F     BA  02D6  1389              POPR    #^M<R3,R4,R5,R6,R7,R8,R9>
00000000'GF         17  02DA  1390              JMP     G^EXE$QIORETURN
                        02E0  1391
            014F    30  02E0  1392 40$:         BSBW    COM_XMITBISYNC                          ; Call BISYNC FDT routine
        03F8 8F     BA  02E3  1393              POPR    #^M<R3,R4,R5,R6,R7,R8,R9>
00000000'GF         17  02E7  1394              JMP     G^EXE$QIORETURN
                        02ED  1395
```

```
                        02ED  1397          .SBTTL  COM_XMITFDT - Common transmit FDT routine
                        02ED  1398 ;COM_XMITFDT -  Common transmit FDT routine
                        02ED  1399 ;
                        02ED  1400 ; The allocated buffer is given to DDCMP where a header is added and if
                        02ED  1401 ; possible the buffer is added to the transmit queue. The buffer keeps various
                        02ED  1402 ; information on the transmit as it progresses from the transmit to the
                        02ED  1403 ; completion queues.
                        02ED  1404 ;
                        02ED  1405 ;       INPUTS
                        02ED  1406 ;               R3 = IRP address
                        02ED  1407 ;               R5 = UCB address
                        02ED  1408 ;               R7 = Address of the user/"Internal" IRP buffer
                        02ED  1409 ;               R8 = Address of allocated buffer
                        02ED  1410 ;               R9 = Size of user/"Internal" IRP buffer
                        02ED  1411 ;
                        02ED  1412 ;       IPL = Fork IPL
                        02ED  1413 ;
                        02ED  1414 ;       OUTPUTS:
                        02ED  1415 ;               R0 = Status of operation
                        02ED  1416 ;               R5,R8 are preserved
                        02ED  1417 ;
                        02ED  1418 COM_XMITFDT:
                  55    02ED  1419          PUSHL   R5                              ; Save these registers
          55    00D0 C5 02EF  1420          MOVL    UCB$A_XG_PRO_BUFFER(R5),R5      ; Get addr of start of TFB
                  42   13 02F4  1421          BEQL    45$                             ; If EQL device not active
              56  02   9A 02F6  1422          MOVZBL  #DLK$C_XMTMSG,R6                ; Set that this is a msg to XMT
                  57   D4 02F9  1423          CLRL    R7                              ; Clear error bits
                 FD02'  30 02FB  1424          BSBW    DDCMP                           ; Branch to set up the header
                  55 8ED0 02FE  1425          POPL    R5                              ; Restore the registers
              56  09   91 0301  1426          CMPB    #DLK$C_ACTNOTCOM,R6             ; If EQL then protocol has not
                  32   13 0304  1427          BEQL    45$                             ; been started
           22 57  04   E0 0306  1428          BBS     #DLK$V_XMTERR,R7,35$            ; If BS then problem with XMT
           03 57  0A   E1 030A  1429          BBC     #DLK$V_XMTCMP,R7,20$            ; If BC no XMT's to complete
                 19E4  30 030E  1430          BSBW    FINISH_XMT_IO                   ; Complete all XMT's
        1C 57  01   E0 0311  1431 20$:       BBS     #DLK$V_PRSTERR,R7,40$           ; Branch BS persisent error
    57  0220 8F   B3 0315  1432          BITW    #<DLK$M_XMTACK!-                ; If EQL then XMT not put on
                       031A  1433                  DLK$M_QFULERR>,R7               ; either queue so abort it
                  10   13 031A  1434          BEQL    35$
              0120 C5   95 031C  1435          TSTB    UCB$B_XG_XSTATE(R5)             ; If NEQ then XMT'r is going
                  05   12 0320  1436          BNEQ    30$
                  51   D4 0322  1437          CLRL    R1                              ; Set no status for start_transmit
                 0982  30 0324  1438          BSBW    START_TRANSMIT                  ; Else start the xmt'r
              50  01   3C 0327  1439 30$:     MOVZWL  S^#SS$_NORMAL,R0                ; Set normal return
                  11   11 032A  1440          BRB     50$
                       032C  1441
              50  14   3C 032C  1442 35$:     MOVZWL  S^#SS$_BADPARAM,R0              ; Set status
                  0C   11 032F  1443          BRB     50$
                       0331  1444
                       0331  1445 40$:     SETBIT  #XM$V_ERR_FATAL,UCB$L_DEVDEPEND(R5) ; Set fatal error
                       0336  1446
                       0336  1447          .IF     DF ERR$$$
                       0336  1448          incw    ucb$w_xg_prst(r5)               ; inc persitant error count
                       0336  1449          .ENDC   ;DF ERR$$$
                       0336  1450
                  EF   11 0336  1451          BRB     30$                             ; Complete as normal timer
                       0338  1452                                                  ; will shutdown the device
    50  20D4 8F   3C 0338  1453 45$:     MOVZWL  #SS$_DEVINACT,R0                ; Set the device is not active
```

```
                51    44 A5    D0  033D  1454 50$:     MOVL    UCB$L_DEVDEPEND(R5),R1          ; Set status for abort
                                 05  0341  1455         RSB
                                     0342  1456
                                     0342  1457 ;COP_BUFF
                                     0342  1458 ;
                                     0342  1459 ; This routine takes the user or "Internal" IRP buffer which contains the
                                     0342  1460 ; message to send and writes it into the allocated buffer.
                                     0342  1461 ;
                                     0342  1462 ;       INPUTS
                                     0342  1463 ;               R1 = Size of allocated buffer
                                     0342  1464 ;               R3 = IRP address
                                     0342  1465 ;               R8 = Address of allocated buffer
                                     0342  1466 ;               R7 = Address of buffer from which to move data
                                     0342  1467 ;               R9 = Size of user/"Internal" IRP buffer
                                     0342  1468 ;
                                     0342  1469 ;               IPL = FIPL from ALT_ENTRY and ASTDEL from XMITFDT
                                     0342  1470 ;
                                     0342  1471 ;       OUTPUTS
                                     0342  1472 ;               Buffer is copied into allocated buffer
                                     0342  1473 ;               R5,R8,R9 are preserved
                                     0342  1474 ;
                                     0342  1475 COP_BUFF:
          08 A8    51    B0  0342  1476         MOVW    R1,XMTQ$W_BUFLEN(R8)            ; Save the size of the buffer
                   13    90  0346  1477         MOVB    S^#DYN$C_BUFIO,-                ; Set that this is an XMT
                0A A8        0348  1478                 XMTQ$B_BUFTYP(R8)
          0C A8    53    D0  034A  1479         MOVL    R3,XMTQ$L_IRP(R8)               ; Save address of the IRP
    1A A8    59    06    A1  034E  1480         ADDW3   #MFD$K_LENGTH,R9,-              ; Get the msg size plus header
                                 0353  1481                 XMTQ$W_MSGSIZE(R8)              ; for character count
                55    DD  0353  1482         PUSHL   R5                              ; Save R5 before the MOV
    2A A8    67    59    28  0355  1483         MOVC    R9,(R7),XMTQ$K_LENGTH(R8)       ; Move data into system buffer
                55 8ED0      035A  1484         POPL    R5                              ; Restore R5
                             05  035D  1485         RSB
```

```
                                035E  1487          .SBTTL  ALT_ENTRY - Alternate I/O entry
                                035E  1488 ;++
                                035E  1489 ; ALT_ENTRY - Alternate I/O entry point
                                035E  1490 ;
                                035E  1491 ; This routine is called by the other drivers to pass an "internal" I/O
                                035E  1492 ; request to the driver.  "Internal" IRP's are not built via $QIO.
                                035E  1493 ; The action here is to setup the IRP fields as if the packet had been
                                035E  1494 ; processed by the FDT routines.
                                035E  1495 ;
                                035E  1496 ; In this driver, the alternate entry point is called by the DECnet
                                035E  1497 ; Transport layer driver.
                                035E  1498 ;
                                035E  1499 ; INPUTS:
                                035E  1500 ;
                                035E  1501 ;         R3 = IRP address
                                035E  1502 ;         R5 = UCB address
                                035E  1503 ;
                                035E  1504 ;         IPL = Fork IPL
                                035E  1505 ;
                                035E  1506 ;         All pertinent fields of the IRP are assumed to be valid.
                                035E  1507 ;
                                035E  1508 ; OUTPUTS:
                                035E  1509 ;
                                035E  1510 ;        R0 = Status of the request
                                035E  1511 ;
                                035E  1512 ;        R3 and R5 preserved
                                035E  1513 ;--
                                035E  1514 ALT_ENTRY:                                   ; Accept an "internal" IRP
                                035E  1515          .IF     DF JNX$$$
                                035E  1516          bsbw    fillbuffer
                                035E  1517          .ENDC   ;DF jnx$$$
                                035E  1518
38 A3    20D4 8F    3C          035E  1519          MOVZWL  #SS$_DEVINACT,IRP$L_MEDIA(R3)   ; Assume device inactive
                    0B    E0    0364  1520          BBS     #XM$V_STS_ACTIVE,-              ; If BS status active
                                0366  1521                  UCB$L_DEVDEPEND(R5),5$          ; device is on line
                  03 44 A5      0369  1522          BRW     IO_DONE                         ; Else comp request in error
50    0167 C5    9A            036C  1523 5$:      MOVZBL  UCB$B_XG_PROTYPE(R5),R0         ; Get protocol type
                                0371  1524          CASE    R0,TYPE=B,<-                    ; Case on protocol type
                                0371  1525                  25$,-                           ; LAPB
                                0371  1526                  27$,-                           ; BISYNC
                                0371  1527                  >                               ; Fall thru on DDCMP
                    01    E0    0379  1528          BBS     #IRP$V_FUNC,-                   ; If BS then receive function
                  5B 2A A3      037B  1529                  IRP$W_STS(R3),ALT_RCVFDT
                   03F8 8F      037E  1530          PUSHR   #^M<R3,R4,R5,R6,R7,R8,R9>       ; Save registers
                59    32 A3     0382  1531          MOVZWL  IRP$W_BCNT(R3),R9               ; Get the length of the buffer
                    33    13    0386  1532          BEQL    10$                             ; If EQL then bad parameter
                  42 A5   59    0388  1533          CMPW    R9,UCB$W_DEVBUFSIZ(R5)          ; If GTR then report as error
                    2D    1A    038C  1534          BGTRU   10$
51    2A         59    C1      038E  1535          ADDL3   R9,#XMTQ$K_LENGTH,R1            ; Add the XMTQ length
                    56  53      0392  1536          MOVL    R3,R6                           ; Save the IRP
                00000000'GF     0395  1537          JSB     G^EXE$ALONONPAGED               ; Allocated the buffer
                    21  50      039B  1538          BLBC    R0,20$                          ; If LBC not buffer allocated
                    53  56      039E  1539          MOVL    R6,R3                           ; Retreive the IRP
                    58  52      03A1  1540          MOVL    R2,R8                           ; Get buffer address in R8
                      80 8F     03A4  1541          MOVB    #XMTQ$M_INTERNAL,-              ; Set that the I/O is from
                      1F A8     03A7  1542                  XMTQ$B_FLAG(R8)                 ; an "Internal" IRP
                                03A9  1543
```

```
                        03A9  1544          .IF     DF JNX$$$
                        03A9  1545          movl    r2,irp$L_fr4(r3)
                        03A9  1546          .ENDC   ;Df jnx$$$
                        03A9  1547
57   2C B3    D0        03A9  1548          MOVL    @IRP$L_SVAPTE(R3),R7            ; Get interal IRP buffer addr
       FF92    30        03AD  1549          BSBW    COP_BUFF                        ; Copy buffer into allcd buff
       FF3A    30        03B0  1550          BSBW    COM_XMITFDT                     ; Branch for common processng
     15 50     E9        03B3  1551          BLBC    R0,30$                          ; If BC complete in error
  03F8 8F      BA        03B6  1552          POPR    #^M<R3,R4,R5,R6,R7,R8,R9>       ; Restore registers
               05        03BA  1553          RSB
                        03BB  1554
38 A3    14    3C        03BB  1555 10$:     MOVZWL  S^#SS$_BADPARAM,IRP$L_MEDIA(R3) ; Set abort status
  03F8 8F      BA        03BF  1556 20$:     POPR    #^M<R3,R4,R5,R6,R7,R8,R9>       ; Restore registers
       18E5    31        03C3  1557          BRW     IO_DONE                         ; Complete the request
                        03C6  1558
         2B    11        03C6  1559 25$:     BRB     ALT_ENTRY_LAPB                  ; Branch to LAPB support
       0028    31        03C8  1560 27$:     BRW     ALT_ENTRY_BISYNC                ; Branch to BISYNC support
                        03CB  1561
   51    58    D0        03CB  1562 30$:     MOVL    R8,R1                           ; Save R8 it contains XMTQ buff
  03F8 8F      BA        03CE  1563          POPR    #^M<R3,R4,R5,R6,R7,R8,R9>       ; Restore registers
38 A3    50    D0        03D2  1564          MOVL    R0,IRP$L_MEDIA(R3)              ; Set status
       18C0    31        03D6  1565          BRW     TRANSMIT_IO_DONE                ; Complete deall the buffer
                        03D9  1566
                        03D9  1567 ALT_RCVFDT:
                        03D9  1568 RCV[APB:
52   2C A3    D0        03D9  1569          MOVL    IRP$L_SVAPTE(R3),R2             ; Is there a buffer to reuse
         06    13        03DD  1570          BEQL    10$                             ; If EQL then no
      2C A3    D4        03DF  1571          CLRL    IRP$L_SVAPTE(R3)                ; Clear so not deallocated
       1242    30        03E2  1572          BSBW    ADDFREELIST                     ; Else add it to the free list
       013A    30        03E5  1573 10$:     BSBW    COM_RCVFDT                      ; Do common processing
     01 50     E9        03E8  1574          BLBC    R0,20$                          ; Br if unsuccessful
               05        03EB  1575          RSB
                        03EC  1576
38 A3    50    D0        03EC  1577 20$:     MOVL    R0,IRP$L_MEDIA(R3)              ;  et status
       1888    31        03F0  1578          BRW     IO_DONE                         ; Complete IRP in error
                        03F3  1579 ;++
                        03F3  1580 ;
                        03F3  1581 ; This routine is the alt entry point for the device when running LAPB or BISYNC
                        03F3  1582 ; mode. Instead of simply checking the function bit in STS it must explicity
                        03F3  1583 ; check the function code in FUNC. It processes READ's ,WRITE's and CLEAN's
                        03F3  1584 ; via this interface. This routine makes some assumptions. First that the
                        03F3  1585 ; interface will only take Logical read's and write's. Second that the
                        03F3  1586 ; io fields will not change.
                        03F3  1587 ;
                        03F3  1588 ;       NOTE:   When looking at this ASSUME statement, pleaser remember that
                        03F3  1589 ;               the values specified are in decimal not in HEX.
                        03F3  1590 ;--
                        03F3  1591 ALT_ENTRY_LAPB:
                        03F3  1592 ALT_ENTRY_BISYNC:
                        03F3  1593          ASSUME  IO$_WRITELBLK EQ 32
                        03F3  1594          ASSUME  IO$_READLBLK EQ 33
                        03F3  1595          ASSUME  IO$_CLEAN EQ 30
                        03F3  1596
50   20 A3    9A        03F3  1597          MOVZBL  IRP$W_FUNC(R3),R0               ; Get the function code
50   FC 8F    8A        03F7  1598          BICB    #^XFC,R0                        ; clear all but the low 2 bits
                        03FB  1599          CASE    R0,TYPE=B,<-                    ; Do the case off this
                        03FB  1600                  XMITLAPB,-
```

```
                03FB  1601                      RCVLAPB>
                0403  1602 ;                    ALT_CLEANFDT>                   ; Fall thru on Clean
                0403  1603
```

```
                     0403  1605          .SBTTL  CLEANFDT/ALT_CLEANFDT - Perform CLEAN FDT processing
                     0403  1606 ;++
                     0403  1607 ;CLEANFDT/ALT_CLEANFDT  - Perform CLEAN FDT processing
                     0403  1608 ;
                     0403  1609 ; This routine sets up to ABORT all outstanding XMT's. This function is
                     0403  1610 ; valid only when the device is running LAPB.
                     0403  1611 ;
                     0403  1612 ;       Inputs:
                     0403  1613 ;
                     0403  1614 ;       R3 = IRP address
                     0403  1615 ;       R4 = PCB address
                     0403  1616 ;       R5 = UCB address
                     0403  1617 ;
                     0403  1618 ;       ALT_CLEANFDT only:
                     0403  1619 ;
                     0403  1620 ;       IPL = FIPL
                     0403  1621 ;
                     0403  1622 ;       Outputs:
                     0403  1623 ;
                     0403  1624 ;       R0 = status return for request
                     0403  1625 ;
                     0403  1626 ;       R3-R5 are preserved.
                     0403  1627 ;
                     0403  1628 ;--
                     0403  1629 ALT_CLEANFDT:
03 68 A5    0F   E0  0403  1630          BBS     #XG_DS_V_CLEAN,UCB$W_DEVSTS(R5),10$ ; If BS then don't allow
        0B49     31  0408  1631          BRW     CLEAN                               ; another clean to happen
                     040B  1632
   38 A3    2C   3C  040B  1633 10$:     MOVZWL  #SS$_ABORT,IRP$L_MEDIA(R3)          ; Set abort status
        1899     31  040F  1634          BRW     IO_DONE                             ; Complete the request in err
                     0412  1635 CLEANFDT:
15 68 A5    0F   E0  0412  1636          BBS     #XG_DS_V_CLEAN,UCB$W_DEVSTS(R5),10$ ; If BS then don't allow
     03F8 8F   BB  0417  1637          PUSHR   #^M<R3,R4,R5,R6,R7,R8,R9>           ; Save registers
                     041B  1638          SETIPL  UCB$B_FIPL(R5)                      ; Set IPL to fork
        0B32     30  041F  1639          BSBW    CLEAN                               ; Do the Clean processing
     03F8 8F   BA  0422  1640          POPR    #^M<R3,R4,R5,R6,R7,R8,R9>
00000000'GF   17  0426  1641          JMP     G^EXE$QIORETURN                     ; Return from processing
                     042C  1642                                                      ; IRP will get posted thru
                     042C  1643                                                      ; IO_DONE
00000000'GF   17  042C  1644 10$:     JMP     G^EXE$ABORTIO                       ; If clean in progress abort
                     0432  1645                                                      ; any others that may be issued
                     0432  1646
```

```
                                0432  1648          .SBTTL  XMITLAPB - Transmit LAPB/BISYNC FDT routine
                                0432  1649
                                0432  1650 ;++
                                0432  1651 ; XMITLAPB - Special LAPB/BISYNC transmit routine
                                0432  1652 ;
                                0432  1653 ; This routine is called after a transmit buffer has been allocated
                                0432  1654 ; and the transmit is to be given to the device. It's called
                                0432  1655 ; to get around the special checks done for DDCMP transmits.
                                0432  1656 ;
                                0432  1657 ;       INPUTS  R1 = Size of allocated buffer
                                0432  1658 ;               R3 = IRP address
                                0432  1659 ;               R5 = UCB address
                                0432  1660 ;               R7 = Address of user or "Internal" IRP buffer
                                0432  1661 ;               R8 = Address of allocated buffer
                                0432  1662 ;
                                0432  1663 ;       OUTPUTS Transmit given to the board
                                0432  1664 ;
                                0432  1665 ;       *** NOTE ***
                                0432  1666 ;
                                0432  1667 ; All messages for LAPB get inserted on an XMTQ, which sits at the same offset
                                0432  1668 ; as DDCMP's CTLQ. This  is done to decrease the overhead involved on the queue
                                0432  1669 ; management. Notice  however that the bit indicating that the message was a
                                0432  1670 ; control message is  never set in the block. This means that when the interrupt
                                0432  1671 ; for the message  is gotten the message will be complete via FORK DONE and
                                0432  1672 ; eventually given  to POST instead of being dropped as normal DDCMP control
                                0432  1673 ; messages are
                                0432  1674 ; are dropped.
                                0432  1675 ;
                                0432  1676 ; These routines are also called when the driver is running in BISYNC mode.
                                0432  1677 ; The handling of tranmits for the two protocols are very similar, therefor
                                0432  1678 ; they share this code. There is a difference in that the BISYNC  messages
                                0432  1679 ; and the LAPB messages are queued to different queues. Bisync mode can not
                                0432  1680 ; take advantage of being queued to the DDCMP CTLQ because it runs
                                0432  1681 ; in half duplex. This means that the protocol must some times must wait
                                0432  1682 ; for CTS to come high before transmitting. The way the transmit code works
                                0432  1683 ; is that is gets the first message off the CTLQ and if empty then off the XMTQ.
                                0432  1684 ; If it can send the message then it will otherwise it must put the message
                                0432  1685 ; back on a queue according to the XMTQ$B_FLAG's
                                0432  1686 ; CONTROL bit is set. This bit also governs whether or not the message
                                0432  1687 ; is complete via IOPOST. Since the BISYNC XMT message may be put on
                                0432  1688 ; the queue and must go thru IOPOST processing we can not use the same
                                0432  1689 ; snort cut used to queue LAPB XMT messages. Thus BISYNC messages are queued
                                0432  1690 ; to the DDCMP XMTQ, which is the second of the queues. The side effect is to
                                0432  1691 ; increase the overhead slightly, but BISYNC is not destined to be a high
                                0432  1692 ; speed prototcol
                                0432  1693 ;
                                0432  1694          .ENABL  LSB
                                0432  1695 COM_XMITLAPB:
                                0432  1696 COM_XMITBISYNC:
     03F8 8F    BB              0432  1697          PUSHR   #^M<R3,R4,R5,R6,R7,R8,R9>       ; Save registers
          41    11              0436  1698          BRB     5$
                                0438  1699 XMITLAPB:
     03F8 8F    BB              0438  1700          PUSHR   #^M<R3,R4,R5,R6,R7,R8,R9>       ; Save registers
0167 C5   01    91              043C  1701          CMPB    #XG$C_PROTYPE_BISYNC,UCB$B_XG_PROTYPE(R5) ; BISYNC?
          0B    12              0441  1702          BNEQ    3$                              ; Branch if not
  50  40 A3    D0              0443  1703          MOVL    IRP$L_ABCNT(R3),R0              ; Should we update the state info
          05    13              0447  1704          BEQL    3$                              ; If EQL then don't
```

```
                015C C5   60    7D  0449  1705          MOVQ    (R0),UCB$Q_XG_STATE_INFO(R5)    ; Update the state info
                  59    32 A3   3C  044E  1706 3$:      MOVZWL  IRP$W_BCNT(R3),R9               ; Get the length of the buffer
                             76   13  0452  1707          BEQL    30$                             ; If EQL then bad parameter
                            42 A5 59   B1  0454  1708    CMPW    R9,UCB$W_DEVBUFSIZ(R5)          ; If GTR then report as error
                             70   1A  0458  1709          BGTRU   30$
                          51   2A   59   C1  045A  1710    ADDL3   R9,#XMTQ$K_LENGTH,R1            ; Add the XMTQ length
                               56   53   D0  045E  1711    MOVL    R3,R6                           ; Save the IRP
                           00000000'GF   16  0461  1712    JSB     G^EXE$ALONONPAGED               ; Allocated the buffer
                                 64   50   E9  0467  1713    BLBC    R0,40$                          ; If LBC not buffer allocated
                                    53   56   D0  046A  1714    MOVL    R6,R3                           ; Retreive the IRP
                                    58   52   D0  046D  1715    MOVL    R2,R8                           ; Get buffer address in R8
                                      80 8F   90  0470  1716    MOVB    #XMTQ$M_INTERNAL,-              ; Set that the I/O is from
                                      1F A8       0473  1717                                            XMTQ$B_FLAG(R8)                 ; an "Internal" IRP
                            57    2C B3   D0  0475  1718    MOVL    @IRP$L_SVAPTE(R3),R7            ; Get interal IRP buffer addr
                                          0479  1719
                              08 A8   51   B0  0479  1720 5$:      MOVW    R1,XMTQ$W_BUFLEN(R8)            ; Save the size of the buffer
                              0A A8   13   90  047D  1721          MOVB    S^#DYN$C_BUFIO,XMTQ$B_BUFTYP(R8) ; Set that buffer is an XMT
                              0C A8   53   D0  0481  1722          MOVL    R3,XMTQ$L_IRP(R8)               ; Save address on the IRP
                              1A A8   59   B0  0485  1723          MOVW    R9,XMTQ$W_MSGSIZE(R8)           ; Set message size
                                      55   DD  0489  1724          PUSHL   R5
                        24 A8   67    59   28  048B  1725          MOVC    R9,(R7),XMTQ$B_MSGHDR(R8)       ; Copy the msg to send into buf
                                      55 8ED0  0490  1726          POPL    R5
                           04 1F A8   07   E0  0493  1727          BBS     #XMTQ$V_INTERNAL,XMTQ$B_FLAG(R8),10$ ; If BS then  at FIPL
                                          0498  1728          SETIPL  UCB$B_FIPL(R5)                  ; Else set to FIPL
                           34 44 A5   0B   E1  049C  1729 10$:     BBC     #XM$V_STS_ACTIVE,UCB$L_DEVDEPEND(R5),50$ ; If BC device inactv
                            54   00D0 C5   D0  04A1  1730          MOVL    UCB$A_XG_PRO_BUFFER(R5),R4      ; Get protocol buffer
                                      2D   13  04A6  1731          BEQL    50$                             ; If eql then dev inactive
                             0167 C5   01   91  04A8  1732          CMPB    #XG$C_PROTYPE_BISYNC,UCB$B_XG_PROTYPE(R5); If EQL then LAPB
                                      33   13  04AD  1733          BEQL    60$
                               24 B4   68   0E  04AF  1734          INSQUE  (R8),@LAPB$Q_XMTQ+4(R4)         ; Insert XMT at end of queue
                                  0120 C5   95  04B3  1735 14$:     TSTB    UCB$B_XG_XSTATE(R5)             ; If NEQL then XMT'r is busy
                                      05   12  04B7  1736          BNEQ    20$
                                      51   D4  04B9  1737          CLRL    R1                              ; Set no status for start transmit
                                    07EB   30  04BB  1738          BSBW    START_TRANSMIT                  ; Else start the transmit
                                 50   01   3C  04BE  1739 20$:     MOVZWL  S^#SS$_NORMAL,R0                ; Set for normal return
                               51   44 A5   D0  04C1  1740          MOVL    UCB$L_DEVDEPEND(R5),R1          ; Set device depend field
                                  03F8 8F   BA  04C5  1741          POPR    #^M<R3,R4,R5,R6,R7,R8,R9>       ; Restore registers
                                      05   04  04C9  1742          RSB
                                           04CA  1743
                              38 A3   14   3C  04CA  1744 30$:     MOVZWL  S^#SS$_BADPARAM,IRP$L_MEDIA(R3) ; Set abort status
                                  03F8 8F   BA  04CE  1745 40$:     POPR    #^M<R3,R4,R5,R6,R7,R8,R9>       ; Restore registers
                                     17D6   31  04D2  1746          BRW     IO_DONE                         ; Complete the request
                                  03F8 8F   BA  04D5  1747 50$:     POPR    #^M<R3,R4,R5,R6,R7,R8,R9>       ; Restore registers
                        38 A3     20D4 8F   3C  04D9  1748          MOVZWL  #SS$_DEVINACT,IRP$L_MEDIA(R3)   ; Assume device inactive
                                     17B7   31  04DF  1749          BRW     TRANSMIT_IO_DONE                ; Complete the request in error
                                            04E2  1750 ;
                                            04E2  1751 ; If the device is running BISYNC half duplex mode the set the SELECT
                                            04E2  1752 ; flag in the the field XMTQ$B_FLAG so that RTS .s dropped after each
                                            04E2  1753 ; message
                                            04E2  1754 ;
                           05 44 A5   02   E1  04E2  1755 60$:     BBC     #XM$V_CHR_HDPLX,UCB$L_DEVDEPEND(R5),65$ ; IF BC not half duplex
                                            04E7  1756          SETBIT  #XMTQ$V_SELECT,XMTQ$B_FLAG(R8)  ; Set select in XMT buffer
                               34 B4   68   0E  04EC  1757 65$:     INSQUE  (R8),@BISYNC$Q_XMTQ+4(R4)       ; Insert XMT at end of queue
                                      C1   11  04F0  1758          BRB     14$
                                           04F2  1759
                                           04F2  1760          .DSABL  LSB
```

```
                                        .SBTTL  RCVFDT - Receive I/O FDT routine
                                ;++
                                ; RCVFDT - Receive I/O FDT routine
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ; The specified buffer is checked for accessibility. The buffer address and
                                ; count are saved in the packet. Then IPL is set to device fork IPL and if
                                ; a message is available the operation is completed.  Otherwise the packet
                                ; is queued onto the waiting receive list.
                                ;
                                ; For requests specifing IO$M_NOW, the I/O is completed with status of
                                ; SS$_ENDOFILE if no message is available when the test is made.
                                ;
                                ;
                                ; INPUTS:
                                ;
                                ;       R3 = I/O packet address
                                ;       R4 = PCB address
                                ;       R5 = UCB address
                                ;       R6 = CCB address
                                ;       R7 = Function code
                                ;       AP = Address of first I/O request parameter
                                ;
                                ; OUTPUTS:
                                ;
                                ;       R0 = Status of the receive request
                                ;
                                ;       R3-R7 preserved.
                                ;--
                                RCVFDT:                                         ; Receive function routine
    50    14    3C              MOVZWL  #SS$_BADPARAM,R0                        ; Assume illegal size
  51    04 AC   3C              MOVZWL  P2(AP),R1                               ; Get size
        1F      13              BEQL    ABORTIO                                 ; Br if none specified
    50    6C    D0              MOVL    P1(AP),R0                               ; Get buffer address
 38 A3    50    D0              MOVL    R0,IRP$L_MEDIA(R3)                      ; Save address
      30 A3     B4              CLRW    IRP$W_BOFF(R3)                          ; No quota to return during
                                                                                ; completion
00000000'GF     16              JSB     G^EXE$READCHK                           ; Check buffer accessibility
                                                                                ; (no return on no access)
                                        SETIPL  UCB$B_FIPL(R5)                  ; Synchronize access to the UCB
        11      10              BSBB    COM_RCVFDT                              ; Process the request
      06 50     E9              BLBC    R0,ABORTIO                              ; Br if error
00000000'GF     17              JMP     G^EXE$QIORETURN                         ; Return to await completion
                                ABORTIO:                                        ; Abort the I/O request
        51      D4              CLRL    R1                                      ; Don't return device status
00000000'GF     17              JMP     G^EXE$ABORTIO                           ;
                                ;
                                ; Common receive processing
                                ;
                                COM_RCVFDT:                                     ; Common receive processing
        0B      E0              BBS     #XM$V_STS_ACTIVE,-                      ; Br if device active
    06 44 A5                            UCB$L_DEVDEPEND(R5),10$
50    2004 8F   3C              MOVZWL  #SS$_DEVINACT,R0                        ; Set return status
                05              RSB
```

```
                          052D  1819 ;
                          052D  1820 ; Check for an avaiable message and complete the receive
                          052D  1821 ;
                          052D  1822
      52   00D4 D5    0F  052D  1823 10$:    REMQUE  @UCB$Q_XG_ATTN(R5),R2           ; Dequeue a received message
                   10    1D  0532  1824          BVS     15$                     ; Br if none
      50   0167 C5    9A  0534  1825          MOVZBL  UCB$B_XG_PROTYPE(R5),R0 ; Get protocol type
                           0539  1826          CASE    R0,TYPE=B,<-            ; Case on protocol type
                           0539  1827                  30$,-                   ; LAPB
                           0539  1828                  30$,-                   ; BISYNC
                           0539  1829                  >                       ; Fall thru on DDCMP
            1710   31      0541  1830          BRW     FINISH_RCV_IO           ; Comp the I/O request
                          0544  1831 ;
                          0544  1832 ; Queue the requst for future message arrival unless IO$M_NOW specified.
                          0544  1833 ; Receives are queued to the special receive wait queue.
                          0544  1834 ;
   09 20 A3    06    E0  0544  1835 15$:    BBS     #IO$V_NOW,IRP$W_FUNC(R3),20$    ; Br BS read NOW
    00E0 D5    63    0E  0549  1836          INSQUE  (R3),@UCB$Q_XG_RCVS+4(R5)       ; Queue the I/O packet
          50   01    3C  054E  1837          MOVZWL  S^#SS$_NORMAL,R0                ; Set QIO status
               05    05  0551  1838          RSB                             ;
                          0552  1839
38 A3   0870 8F    3C  0552  1840 20$:    MOVZWL  #SS$_ENDOFFILE,IRP$L_MEDIA(R3)  ; Set no message status
            1750   30  0558  1841          BSBW    IO_DONE                 ; Complete the I/O
          50   01    3C  055B  1842          MOVZWL  S^#SS$_NORMAL,R0                ; Set normal completetion
               05    05  055E  1843          RSB                             ; And return
                          055F  1844 ;
                          055F  1845 ; The RCV finish IO routines are the same for both
                          055F  1846 ; the LAPB and BISYNC protocols
                          055F  1847 ;
      62   42 A2    DE  055F  1848 30$:    MOVAL   RCV_Z_HEADER(R2),(R2)           ; Set up point to data
      50   0C A2    3C  0563  1849          MOVZWL  RCV_W_MSGSIZ(R2),R0             ; Get size of transfer
            16F2   31      0567  1850          BRW     FINISH_RCV_IO_LAPB              ; Go thru LAPB finish IO
                          056A  1851
```

```
                        .SBTTL  SETMODEFDT,  Set mode I/O operation FDT routine

;++
; SETMODEFDT - Set mode I/O operation FDT routine
;
; Functional description:
;
; This routine is used to set the configuration of the DMF32 hardware device
; including configuration of software DDCMP. Subfunction modifier bits are used
; to specify the type of action to be taken. The two characteristics buffers
; (P1 and P2) are used to describe specific characteristics.
;
; The QIO parameters for SETMODE are:
;
;       P1 = Optional address of quadword or longword buffer
;       P2 = Optional address of buffer descriptor for extended characteristics
;       P3 = Number of receive buffers to pre-allocate. Required on
;            controller startup.
;
;
; The subfunction modifiers are as follows:
;
;
;        o STARTUP - start the device - this modifier is used to
;              start the device.
;
;       o SHUTDOWN - shutdown the device - this modifier
;              is used to stop the device.
;
;       o ATTNAST - request an attention AST - this modifier is used
;              to set up an AST to be delivered when a change of
;              status occurs on this device.
;
;       o CTRL - perform the request on the Controller not the
;              tributary.
;
;       o SET_MODEM - set line unit's mode register.
;
; Inputs:
;
;       R3 = IRP address
;       R4 = PCB address
;       R5 = UCB address
;       R6 = CCB address
;       R7 = Function code
;       AP = address of first QIO parameter
;
; Outputs:
;
;       R0 = status of setmode request
;
;       R3-R5 are preserved.
;
;       R7-R9 = destroyed
;
;--
```

```
                     056A  1910 SETMODEFDT:                                        ; Setmode FDT processing
 57   20 A3   3C     056A  1911          MOVZWL  IRP$W_FUNC(R3),R7                ; Get entire function code
 03 57   09   E1     056E  1912          BBC     #IO$V_CTRL,R7,5$                 ; Br if not controller request
       00AA   31     0572  1913          BRW     SETMODE_CTRL                     ; Process controller request
                     0575  1914 ;
                     0575  1915 ; Perform setmode request on a tributary
                     0575  1916 ;
0167 C5   02   91     0575  1917 5$:      CMPB    #XG$C_PROTYPE_DDCMP,UCB$B_XG_PROTYPE(R5) ; NEQ then abort io
          64   12     057A  1918          BNEQ    40$
 2A 57   08   E1     057C  1919          BBC     #IO$V_ATTNAST,R7,30$             ; Branch if not attention AST
                     0580  1920
                     0580  1921 ;
                     0580  1922 ; User is requesting an attention AST.
                     0580  1923 ;
57   0090 C5   DE     0580  1924          MOVAL   UCB$L_XG_AST(R5),R7              ; Get addr of AST list
 00000000'GF   16     0585  1925          JSB     G^COM$SETATTNAST                 ; Set up attention AST
51   00D4 C5   9E     058B  1926          MOVAB   UCB$Q_XG_ATTN(R5),R1             ; Check for empty rcv list
     61   51   D1     0590  1927          CMPL    R1,(R1)                          ; Empty?
          08   13     0593  1928          BEQL    20$                              ; Yes, no need to inform user
          53   DD     0595  1929 10$:     PUSHL   R3                               ; Else, save IRP address
        1AC1   30     0597  1930          BSBW    POKE_USER                        ; Inform the user
          53 8ED0     059A  1931          POPL    R3                               ; Restore IRP address
 51   44 A5   D0     059D  1932 20$:     MOVL    UCB$L_DEVDEPEND(R5),R1           ; Get device characteristics
    50   01   3C     05A1  1933 23$:     MOVZWL  S^#SS$_NORMAL,R0                 ; Set success
 00000000'GF   17     05A4  1934 25$:     JMP     G^EXE$FINISHIO                   ; Complete the I/O
                     05AA  1935
                     05AA  1936
       04CF   30     05AA  1937 30$:     BSBW    GET_CHAR_BUFS                    ; Get P1 and P2 characteristics
    30 50   E9     05AD  1938          BLBC    R0,40$                           ; Br if error - abort I/O
 32 57   07   E1     05B0  1939          BBC     #IO$V_SHUTDOWN,R7,50$            ; Branch if not trib shutdown
                     05B4  1940 ;
                     05B4  1941 ; Shutdown tributary modifier specified.
                     05B4  1942 ;
                     05B4  1943 ; Validate P2 buffer. Then update trib parameter block.
                     05B4  1944 ;
       02   9C     05B4  1945          MOVB    S^#XG_FC_V_STOP_CIR,-            ; Set internal function code
    21 A3           05B6  1946                  IRP$B_XGFUNC(R3)
52   44 A5   3C     05B8  1947          MOVZWL  UCB$L_DEVDEPEND(R5),R2           ; Else, get status
                     05BC  1948 35$:     DSBINT  UCB$B_FIPL(R5)                   ; Sync to get the UCB
        1DFB   30     05C3  1949          BSBW    VALIDATE_P2_TRIB                 ; Validate the P2 buffer
                     05C6  1950          ENBINT                                   ; Restore IPL
51   44 A5   D0     05C9  1951          MOVL    UCB$L_DEVDEPEND(R5),R1           ; Assume no error
     D4 50   E9     05CD  1952          BLBC    R0,25$                           ; Br if error
                     05D0  1953          DSBINT  UCB$B_FIPL(R5)                   ; Sync to get the UCB
        066E   30     05D7  1954          BSBW    CHG_TRIB                         ; Change trib parameters
                     05DA  1955          ENBINT                                   ; Restore IPL
        048C   31     05DD  1956          BRW     QUEPKT                           ; Queue packet to driver
                     05E0  1957
00000000'GF   17     05E0  1958 40$:     JMP     G^EXE$ABORTIO                    ; Abort the I/O request
                     05E6  1959
52   44 A5   3C     05E6  1960 50$:     MOVZWL  UCB$L_DEVDEPEND(R5),R2           ; Get status
                     05EA  1961          DSBINT  UCB$B_FIPL(R5)                   ; Sync to get the UCB
        1DCD   30     05F1  1962          BSBW    VALIDATE_P2_TRIB                 ; Validate the P2 buffer
                     05F4  1963          ENBINT                                   ; Restore IPL
        E6 50   E9     05F7  1964          BLBC    R0,40$                           ; Br if error
                     05FA  1965 60$:     DSBINT  UCB$B_FIPL(R5)                   ; Sync to get the UCB
        0644   30     0601  1966          BSBW    CHG_TRIB                         ; Change trib parameters
```

```
                                                M 5
                     0604  1967                 ENBINT                                  ; Restore IPL
OD 57    06   E0     0607  1968                 BBS      #IO$V_STARTUP,R7,80$           ; Br if startup request
   50    01   3C     060B  1969                 MOVZWL   S^#SS$_NORMAL,R0               ; Else, set successful return
51   44 A5   D0     060E  1970                 MOVL     UCB$L_DEVDEPEND(R5),R1         ; Set IOSB return status
00000000'GF   17     0612  1971                 JMP      G^EXE$FINISHIO                 ; Finish the I/O request
        00   90     0618  1972 80$:             MOVB     S^#XG_FC_V_STRT_CIR,-          ; Set internal function code
      21 A3         061A  1973                          IRP$B_XGFUNC(R3)
       044D   31     061C  1974                 BRW      QUEPKT
                     061F  1975
```

```
                                061F  1977          .SBTTL  SETMODE_CTRL,  Perform setmode FDT operation on controller
                                061F  1978
                                061F  1979 ;++
                                061F  1980 ; SETMODE_CTRL - Perform setmode FDT operation on controller
                                061F  1981 ;
                                061F  1982 ; Functional description:
                                061F  1983 ;
                                061F  1984 ; This routine performs the SETMODE FDT setup for the controller.
                                061F  1985 ;
                                061F  1986 ; Inputs:
                                061F  1987 ;
                                061F  1988 ;       R3 = IRP address
                                061F  1989 ;       R4 = PCB address
                                061F  1990 ;       R5 = UCB address
                                061F  1991 ;       R7 = IRP function word
                                061F  1992 ;
                                061F  1993 ; Outputs:
                                061F  1994 ;
                                061F  1995 ;       R0 = status of setmode request
                                061F  1996 ;
                                061F  1997 ;       R3-R5 are preserved.
                                061F  1998 ;
                                061F  1999 ;--
                                061F  2000
                                061F  2001 SETMODE_CTRL:                                ; Perform setmode on controller
                         045A   30  061F  2002          BSBW    GET_CHAR_BUFS           ; Get P1 and P2 characteristics
                       7B 50    E9  0622  2003          BLBC    R0,20$                  ; Br if error - Abort I/O
                                    0625  2004
                                B3  0625  2005          BITW    #<XM$M_CHR_CTRL!-       ; If NEQ then not multipoint
                                    0626  2006                  XM$M_CHR_DMC>,-         ; control or DMC mode
                3C A3   0060 8F     0626  2007                  IRP$L_MEDIA+4(R3)
                             05 13  062B  2008          BEQL    5$
                        50   14 3C  062D  2009          MOVZWL  #SS$_BADPARAM,R0        ; Else set bad parameter
                             6E 11  0630  2010          BRB     20$
                                    0632  2011
                                    0632  2012 ; This clear of R6 assumes no protocol was specified. If this is true then
                                    0632  2013 ; the driver assumes that the user wanted DDCMP, and if a P1 buffer
                                    0632  2014 ; was speficied the driver will use the characteristics set in the buffer.
                                    0632  2015 ; The P1 buffer is not valid for BISYNC and LAPB modes, and will not be checked.
                                    0632  2016
                             56 D4  0632  2017 5$:      CLRL    R6                      ; Assume no protocol specified
                    51   0458 8F   3C  0634  2018          MOVZWL  #NMA$C_PCLI_PRO,R1      ; Check P2 buffer for multi
                                    0639  2019          DSBINT  UCB$B_FIPL(R5)          ; Sync to get the UCB
                            1F0D    30  0640  2020          BSBW    UNPACK_P2_BUF           ; point control specification
                                    0643  2021          ENBINT                          ; Restore IPL
                           14 50    E9  0646  2022          BLBC    R0,14$                  ; If BC then no prot specified
                        56    52    D0  0649  2023          MOVL    R2,R6                   ; Set protocol type in R6
                        52    01    91  064C  2024          CMPB    #NMA$C_LINPR_CON,R2     ; If EQL then multipoint contrl
                              07    13  064F  2025          BEQL    10$                     ; specified
                        52    04    91  0651  2026          CMPB    #NMA$C_LINPR_DMC,R2     ; If EQL then DDCMP DMC mode
                              02    13  0654  2027          BEQL    10$                     ; specified
                              05    11  0656  2028          BRB     14$
                        50    14    3C  0658  2029 10$:     MOVZWL  #SS$_BADPARAM,R0        ; Else set bad parameter
                              3D    11  065B  2030          BRB     17$
                    52    68 A5    3C  065D  2031 14$:     MOVZWL  UCB$W_DEVSTS(R5),R2     ; Get device status
                                    0661  2032          DSBINT  UCB$B_FIPL(R5)          ; Sync to get the UCB
                            1D64    30  0668  2033          BSBW    VALIDATE_P2_UCB         ; Validate the P2 buffer
```

```
                     066B  2034          ENBINT                                  ; Restore IPL
          2F 50  E9  066E  2035          BLBC    R0,20$                          ; Br if error
    34 57  07    E1  0671  2036          BBC     #IO$V_SHUTDOWN,R7,30$           ; Br if not shutdown request
                     0675  2037 ;
                     0675  2038 ; Shutdown modifier specified
                     0675  2039 ;
          03   90    0675  2040          MOVB    S^#XG_FC_V_STOP_LIN,-           ; Set internal function code
        21 A3          0677  2041                  IRP$B_XGFUNC(R3)
    56  0150 C5  9A    0679  2042          MOVZBL  UCB$B_XG_PRO(R5),R6             ; Set the protocol type for shutdown
                     067E  2043          DSBINT  UCB$B_FIPL(R5)                  ; Sync to get UCB
            04F0 30    0685  2044          BSBW    CHG_UCB                         ; Update the UCB
                     0688  2045          ENBINT                                  ; Lower IPL
          02   E1    068B  2046          BBC     #XG_DS_V_INITED,-               ; Br if controller not up
       03 68 A5          068D  2047                  UCB$W_DEVSTS(R5),15$            ;
            03D9 31    0690  2048          BRW     QUEPKT                          ; Queue packet to driver
                     0693  2049
     51   44 A5  D0    0693  2050 15$:     MOVL    UCB$L_DEVDEPEND(R5),R1          ; Get IOSB1 return
        50   01  3C    0697  2051          MOVZWL  S^#SS$_NORMAL,R0                ; Set success
    00000000'GF  17    069A  2052 17$:     JMP     G^EXE$FINISHIO                  ; Complete the I/O request
                     06A0  2053
    00000000'GF  17    06A0  2054 20$:     JMP     G^EXE$ABORTIO                   ; Abort the I/O request
                     06A6  2055
            00AC 31    06A6  2056 25$:     BRW     50$                             ; Branch to compl request
                     06A9  2057
        56   09  91    06A9  2058 30$:     CMPB    #NMA$C_LINPR_BSY,R6             ; If BIYSNC mode check privs
                     06AC  2059          IFPRIV  CMKRNL,32$                      ; If priv ok
        50   24  3C    06B2  2060          MOVZWL  #SS$_NOPRIV,R0                  ; else abort the io request
             E9  11    06B5  2061          BRB     20$
     EB 57   06  E1    06B7  2062 32$:     BBC     #IO$V_STARTUP,R7,25$            ; Br if not startup request
             01  90    06BB  2063          MOVB    S^#XG_FC_V_STRT_LIN,-           ; Set internal function code
           21 A3          06BD  2064                  IRP$B_XGFUNC(R3)
      51   08 AC  3C    06BF  2065          MOVZWL  P3(AP),R1                       ; If EQL then P3 not set up
             25  13    06C3  2066          BEQL    35$
             02  E1    06C5  2067          BBC     #XG_DS_V_INITED,-               ; If BC then device not inited
        11 68 A5          06C7  2068                  UCB$W_DEVSTS(R5),33$            ; set number of buffers
    0153 C5  51  91    06CA  2069          CMPB    R1,UCB$B_XG_BFN(R5)             ; Are the buffer nmb's the same
             19  13    06CF  2070          BEQL    35$                             ; Br if yes
    51  0451 8F  3C    06D1  2071          MOVZWL  #NMA$C_PCLI_BFN,R1              ; Set IOSB1 return
        50   14  3C    06D6  2072          MOVZWL  #SS$_BADPARAM,R0                ; Set error return
             BF  11    06D9  2073          BRB     17$                             ; Finish the I/O request
                     06DB  2074 33$:     DSBINT  UCB$B_FIPL(R5)                  ; Sync to get UCB
    0153 C5  51  90    06E2  2075          MOVB    R1,UCB$B_XG_BFN(R5)             ; Store new rcve buffer number
                     06E7  2076          ENBINT                                  ; Restore IPL
                     06EA  2077 ;
                     06EA  2078 ; Set new P1, P2 parameters
                     06EA  2079 ;
             02  E1    06EA  2080 35$:     BBC     #XG_DS_V_INITED,-               ; Br if device not already
          03 68 A5          06EC  2081                  UCB$W_DEVSTS(R5),36$            ; inited
            0089 31    06EF  2082          BRW     70$
                     06F2  2083 36$:     DSBINT  UCB$B_FIPL(R5)                  ; Sync to get UCB
          03 38 A3  E9    06F5  2084          BLBC    IRP$L_MEDIA(R3),37$             ; Br if no P1
             44 A5  94    06FD  2085          CLRB    UCB$L_DEVDEPEND(R5)             ; Clear old characteristics
            0475 30    0700  2086 37$:     BSBW    CHG_UCB                         ; Change the UCB charac
                     0703  2087          ENBINT                                  ; Restore IPL
    51   0153 C5  9A    0706  2088          MOVZBL  UCB$B_XG_BFN(R5),R1             ; Get number of receive buffers
    0167 C5  01  91    070B  2089          CMPB    #XG$C_PROTYPE_BISYNC,UCB$B_XG_PROTYPE(R5) ; If neq then branch
             03  12    0710  2090          BNEQ    38$
```

```
                     0712  2091 ;
                     0712  2092 ; For BISYNC mode add an extra receive buffer. This buffer will be
                     0712  2093 ; permenantly allocated to the device to do the acutal receiving of
                     0712  2094 ; data off the line. The other buffers allocated will be used to
                     0712  2095 ; receive the data after the data has been passed thru the framing routine
                     0712  2096 ;
   51    01    C0    0712  2097          ADDL2   S^#1,R1
 52    3A A3    3C    0715  2098 38$:     MOVZWL  IRP$L_MEDIA+2(R3),R2            ; Get message size from P1
   04  38 A3    E8    0719  2099          BLBS    IRP$L_MEDIA(R3),40$             ; Br if P1 buffer valid
 52    42 A5    3C    071D  2100          MOVZWL  UCB$W_DEVBUFSIZ(R5),R2          ; Else, get buff size from UCB
   50    14    3C    0721  2101 40$:     MOVZWL  #SS$_BADPARAM,R0                ; Assume bad parameter
   52    51    C4    0724  2102          MULL    R1,R2                           ; Compute total needed for buff
         4C    13    0727  2103          BEQL    60$                             ; Br if zero - error
   57    52    3C    0729  2104          MOVZWL  R2,R7                           ; Copy quota
   57    52    D1    072C  2105          CMPL    R2,R7                           ; Overflow?
         44    12    072F  2106          BNEQ    60$                             ; Br if error
         53    DD    0731  2107          PUSHL   R3                              ; Save R3
00000000'GF    16    0733  2108          JSB     G^EXE$BUFQUOPRC                 ; Check caller's quota
         53 8ED0    0739  2109          POPL    R3                              ; Restore R3
   36    50    E9    073C  2110          BLBC    R0,60$                          ; Br if error
 40 A3    57    B0    073F  2111          MOVW    R7,IRP$W_QUOTA(R3)              ; Save quota in packet
50   0080 C4    D0    0743  2112          MOVL    PCB$L_JIB(R4),R0                ; Get JIB address
 20 A0    57    C2    0748  2113          SUBL    R7,JIB$L_BYTCNT(R0)             ; Charge user for rec. bufs
                     074C  2114 ;
                     074C  2115 ; The following call is used to allocate a buffer size dependant on
                     074C  2116 ; the protocol, for use with the various protocols supported. If the
                     074C  2117 ; routine returns with LBC then the buffer could not be allocated and
                     074C  2118 ; the device can not be started.
                     074C  2119 ;
    0057    30       074C  2120          BSBW    ALLOC_PROT_BUFFER
  23 50    E9       074F  2121          BLBC    R0,60$                          ; If LBC then abort startup
   0317    31       0752  2122          BRW     QUEPKT                          ; Queue request to driver
                     0755  2123 ;
                     0755  2124 ; No modifier specified - change controller parameters
                     0755  2125 ;
                     0755  2126 50$:     SETIPL  UCB$B_FIPL(R5)                  ; Sync access to UCB
      02    E0       0759  2127          BBS     #XG_DS_V_INITED,-               ; Br if already inited
 1D 68 A5             075B  2128                  UCB$W_DEVSTS(R5),70$            ;
 03 38 A3    E9       075E  2129          BLBC    IRP$L_MEDIA(R3),53$             ; Br if no P1 buffer
    44 A5    94       0762  2130          CLRB    UCB$L_DEVDEPEND(R5)             ; Clear old UCB characteristics
    0410    30       0765  2131 53$:     BSBW    CHG_UCB                         ; Change UCB parameters
   50    01    3C    0768  2132          MOVZWL  S^#SS$_NORMAL,R0                ; Else, set success
 51    44 A5    D0    076B  2133          MOVL    UCB$L_DEVDEPEND(R5),R1          ; Set IOSB1 return
00000000'GF    17    076F  2134 55$:     JMP     G^EXE$FINISHIO                  ; Finish the I/O request
                     0775  2135
00000000'GF    17    0775  2136 60$:     JMP     G^EXE$ABORTIO                   ; Abort the I/O request
                     077B  2137 ;
                     077B  2138 ; Device already inited - set new parameters and give them to device
                     077B  2139 ;
 0F 38 A3    E9       077B  2140 70$:     BLBC    IRP$L_MEDIA(R3),80$             ; Br if no P1 buffer
    3C A3    91       077F  2141          CMPB    IRP$L_MEDIA+4(R3),-             ; Are characteristics okay?
    44 A5             0782  2142                  UCB$L_DEVDEPEND(R5)
      08    13       0784  2143          BEQL    80$                             ; Yes - let it go
   50    14    3C    0786  2144          MOVZWL  #SS$_BADPARAM,R0                ; Return error
   51    01    CE    0789  2145          MNEGL   S^#1,R1                         ; No specific parameter
      E1    11       078C  2146          BRB     55$                             ; Complete the I/O
                     078E  2147
```

```
                     078E  2148 80$:    SETIPL  UCB$B_FIPL(R5)                  ; Sync to get UCB
          03E3  30   0792  2149          BSBW    CHG_UCB                         ; Change UCB parameters
    50    F0 8F  90   0795  2150          MOVB    #<15@4>,R0                      ; Set only four parameters
       50    01  B0   0799  2151          MOVW    S^#SS$_NORMAL,R0
    51    44 A5  D0   079C  2152          MOVL    UCB$L_DEVDEPEND(R5),R1
00000000'GF  17   07A0  2153          JMP     G^EXE$FINISHIO
```

```
                                07A6  2155          .SBTTL  ALLOC_PROT_BUFFER, Allocate a buffer to be used by the protocol
                                07A6  2156 ;++
                                07A6  2157 ;ALLOC_PROT_BUFFER - Allocate protocol buffer routine
                                07A6  2158 ;
                                07A6  2159 ; This routine allocates the  buffer needed for operation of each of the
                                07A6  2160 ; protocols that the driver supports. By allocating the buffer on startup
                                07A6  2161 ; of the device rather than in the UCB we can eliminate some overhead
                                07A6  2162 ; involved in having the device but using a protocol other than DDCMP
                                07A6  2163 ; over it.
                                07A6  2164 ;
                                07A6  2165 ;       INPUTS  R4 = Address of the PCB
                                07A6  2166 ;
                                07A6  2167 ;       OUTPUTS R2,R3 are preserved
                                07A6  2168 ;
                                07A6  2169 ;--
                                07A6  2170 ALLOC_PROT_BUFFER:
                          0C    BB  07A6  2171          PUSHR   #^M<R2,R3>
              50  0167 C5       9A  07A8  2172          MOVZBL  UCB$B_XG_PROTYPE(R5),R0         ; Get protocol type
                                07AD  2173          CASE    R0,TYPE=B,<-                    ; Case on protocol type
                                07AD  2174                  10$,-                           ; LAPB
                                07AD  2175                  20$,-                           ; BISYNC
                                07AD  2176                  >                               ; Fall thru on DDCMP
                                07B5  2177 ;
                                07B5  2178 ; Alloc DDCMP buffer the TFB and GFB are allocated contiguously
                                07B5  2179 ;
      51  2C  000001C8 8F       C1  07B5  2180          ADDL3   #TF$K_LENGTH,#GF$K_LENGTH,R1
                          0C    11  07BD  2181          BRB     30$
                51        38    D0  07BF  2182 10$:     MOVL    #LAPB$K_LENGTH,R1               ; Set LAPB buffer size
                          07    11  07C2  2183          BRB     30$
           51   00000074 8F    D0  07C4  2184 20$:     MOVL    #BISYNC$K_LENGTH,R1             ; Set BISYNC buffer size
                00000000'GF    16  07CB  2185 30$:     JSB     G^EXE$BUFFRQUOTA                ; Does the user have quota
                     51 50     E9  07D1  2186          BLBC    R0,40$                          ; If LBC no quota
                00000000'GF    16  07D4  2187          JSB     G^EXE$ALLOCBUF                  ; Allocate the buffer
                        48 50  E9  07DA  2188          BLBC    R0,40$                          ; If LBC then not allocated
              50    0080 C4    D0  07DD  2189          MOVL    PCB$L_JIB(R4),R0                ; Get the users JIB
                20 A0    51    C2  07E2  2190          SUBL2   R1,JIB$L_BYTCNT(R0)             ; Subtract the quota
                                07E6  2191          DSBINT  UCB$B_FIPL(R5)                  ; Synch to gain access to UCB
              00D0 C5    52    D0  07ED  2192          MOVL    R2,UCB$A_XG_PRO_BUFFER(R5)      ; Save the buffer address
                                07F2  2193          ENBINT                                  ; Enable interrupts
                08 A2    51    B0  07F5  2194          MOVW    R1,UCB$W_SIZE(R2)               ; Save the size of BUFFER allocated
                0A A2    13    B0  07F9  2195          MOVW    S^#DYN$C_BUFIO,UCB$B_TYPE(R2)   ; Set buffer type
              0167 C5    02    91  07FD  2196          CMPB    #XG$C_PROTYPE_DDCMP,UCB$B_XG_PROTYPE(R5)
                          1E    12  0802  2197          BNEQ    35$
                          3E    BB  0804  2198          PUSHR   #^M<R1,R2,R3,R4,R5>
01C8 C2   2C    00    65    00    2C  0806  2199          MOVC5   #0,(R5),#0,#GF$K_LENGTH,TF$K_LENGTH(R2)
                          3E    BA  080E  2200          POPR    #^M<R1,R2,R3,R4,R5>
            01D1 C2     0B    A5  90  0810  2201          MOVB    UCB$B_FIPL(R5),GF$B_FIPL+TF$K_LENGTH(R2) ; Set fork IPL for DDCMP
            01D2 C2     5E    A5  90  0816  2202          MOVB    UCB$B_DIPL(R5),GF$B_DIPL+TF$K_LENGTH(R2) ; Set device IPL for DDCMP
            4C A2    01C8 C2    DE  081C  2203          MOVAL   TF$K_LENGTH(R2),TF$A_GFB(R2)    ; Set address of global field
                     50    01    3C  0822  2204 35$:     MOVZWL  S^#SS$_NORMAL,R0                ; Set successful return
                           0C    BA  0825  2205 40$:     POPR    #^M<R2,R3>
                           05    05  0827  2206          RSB
                                 0828  2207
                                 0828  2208
```

```
                     0828  2210          .SBTTL  SENSEMODEFDT,  Sense Mode I/O operation FDT routine
                     0828  2211
                     0828  2212 ;++
                     0828  2213 ; SENSEMODEFDT - Sense Mode FDT routine
                     0828  2214 ;
                     0828  2215 ; Functional Description:
                     0828  2216 ;
                     0828  2217 ;  This routine returns information to the caller about the configuration
                     0828  2218 ;  and status of the DMF32 device.  Depending on the function modifier,
                     0828  2219 ;  either the device characteristics, error counters contents are returned.
                     0828  2220 ;
                     0828  2221 ;
                     0828  2222 ;  The QIO parameters for SENSEMODE are:
                     0828  2223 ;
                     0828  2224 ;       P1 = optional address of quadword or longword buffer
                     0828  2225 ;       P2 = optional address of buffer descriptor for extended characteristics
                     0828  2226 ;
                     0828  2227 ;
                     0828  2228 ; Inputs:
                     0828  2229 ;
                     0828  2230 ;       R3 = IRP address
                     0828  2231 ;       R4 = PCB address
                     0828  2232 ;       R5 = UCB address
                     0828  2233 ;       R6 = CCB address
                     0828  2234 ;       R7 = Function code
                     0828  2235 ;       AP = Address of first function-dependent QIO parameter
                     0828  2236 ;
                     0828  2237 ; Outputs:
                     0828  2238 ;
                     0828  2239 ;       R0 = status return of sensemode request
                     0828  2240 ;
                     0828  2241 ;       R3-R5 are preserved.
                     0828  2242 ;
                     0828  2243 ;--
                     0828  2244
                     0828  2245 SENSEMODEFDT:                                   ; Sensemode FDT I/O processing
  57   20 A3   B0    0828  2246          MOVW    IRP$W_FUNC(R3),R7               ; Get entire function code
                     082C  2247
  03 57   09   E1    082C  2248          BBC     #IO$V_CTRL,R7,5$                ; Br if not controller request
       00DB   31    0830  2249          BRW     SENSEMODE_CTRL                  ; Else, process controller req.
0167 C5   02   91    0833  2250 5$:       CMPB    #XG$C_PROTYPE_DDCMP,UCB$B_XG_PROTYPE(R5) ; NEQ then abort io
          7B   12    0838  2251          BNEQ    30$
57   0500 8F   B3    083A  2252          BITW    #<IO$M_RD_COUNT!-               ; Check to see if either bit is
                     083F  2253                  IO$M_CLR_COUNT>,R7              ; Is set
          7A   13    083F  2254          BEQL    40$                             ; If EQL then read parameters
          50   D4    0841  2255          CLRL    R0                              ; Assume clear count
     51   01   3C    0843  2256          MOVZWL  S^#SS$_NORMAL,R1                ; Assume success
  41 57   08   E1    0846  2257          BBC     #IO$V_RD_COUNT,R7,15$           ; If BC then not read count
                     084A  2258 ;
                     084A  2259 ; Read tributary counters - modifier RD_COUNT
                     084A  2260 ;
       0296   30     084A  2261          BSBW    CHECK_P2                        ; Check P2 buffer
     50   14   3C    084D  2262          MOVZWL  #SS$_BADPARAM,R0                ; Assume zero length buffer
  3C A3   51   B0    0850  2263          MOVW    R1,IRP$L_MEDIA+4(R3)            ; Save user size of P2 buffer
          5F   13    0854  2264          BEQL    30$                             ; Br if none
          54   DD    0856  2265          PUSHL   R4
     51   01   3C    0858  2266          MOVZWL  S^#SS$_NORMAL,R1                ; Assume success
```

```
                54    00D0 C5    D0   085B  2267          MOVL    UCB$A_XG_PRO_BUFFER(R5),R4      ; Get prot buffer address
                                 08   12   0860  2268          BNEQ    8$                              ; If NEQ then buffer exists
                  32 A3    00    B0   0862  2269          MOVW    #0,IRP$W_BCNT(R3)               ; Else return no errors read
                                 50   D4   0866  2270          CLRL    R0                              ; And complete the request
                                 37   11   0868  2271          BRB     20$
                                           086A  2272
                  50    1E A4    3C   086A  2273  3$:     MOVZWL  TF$W_TEB(R4),R0                 ; Get size of buffer needed
                  50    3C A3    B1   086E  2274          CMPW    IRP$L_MEDIA+4(R3),R0            ; IF GEQU then buffer ok
                                 09   1E   0872  2275          BGEQU   10$
                51    0601 8F    3C   0874  2276          MOVZWL  #SS$_BUFFEROVF,R1               ; Return partial success
                  50    3C A3    3C   0879  2277          MOVZWL  IRP$L_MEDIA+4(R3),R0            ; Set size of copy
                                 3B   BB   087D  2278  10$:     PUSHR   #^M<R0,R1,R3,R4,R5>             ; Save the registers
          62    018A C4    50   28   087F  2279          MOVC3   R0,TF$K_ERRSRT(R4),(R2)         ; Get the errors
                                 3B   BA   0885  2280          POPR    #^M<R0,R1,R3,R4,R5>
                  16 57    0A   E1   0887  2281          BBC     #IO$V_CLR_COUNT,R7,20$          ; Br if not clear counts
                     57    03   3C   088B  2282  15$:     MOVZWL  #<DLK$M_TRIB!DLK$M_CLEAR>,R7    ; Set counters to read/clear
                                      088E  2283          SETIPL  UCB$B_FIPL(R5)                  ; Set to fork IPL
                                 2B   BB   0892  2284          PUSHR   #^M<R0,R1,R3,R5>                ; Save registers
                                 58   D4   0894  2285          CLRL    R8                              ; Set no parameters to return
                     55    54   D0   0896  2286          MOVL    R4,R5                           ; Set address of trib block
                     56    03   9A   0899  2287          MOVZBL  #DLK$C_REQEBA,R6                ; Set operation to perform
                        F761'   30   089C  2288          BSBW    DDCMP                           ; Branch to protocol
                                 2B   BA   089F  2289          POPR    #^M<R0,R1,R3,R5>
                50    50    10   78   08A1  2290  20$:     ASHL    #16,R0,R0                       ; Shift size of buffer return
                     50    51   B0   08A5  2291          MOVW    R1,R0                           ; Set success status
                                 54   8ED0 08A8  2292          POPL    R4
                  51    44 A5    D0   08AB  2293          MOVL    UCB$L_DEVDEPEND(R5),R1          ; Set devdepend info
                00000000'GF      17   08AF  2294          JMP     G^EXE$FINISHIO                  ; Complete the request
                                           08B5  2295
                00000000'GF      17   08B5  2296  30$:     JMP     G^EXE$ABORTIO                   ; Abort the I/O request
                                           08BB  2297
                                           08BB  2298 ;
                                           08BB  2299 ; Read tributary parameters
                                           08BB  2300 ;
                     51    08   3C   08BB  2301  40$:     MOVZWL  S^#8,R1                         ; Size of P1 buffer if present
                        0220    30   08BE  2302          BSBW    CHECK_BUFS                      ; Check P1 and P2 buffers
                     50    01   3C   08C1  2303          MOVZWL  S^#SS$_NORMAL,R0                ; Assume success
                  3C A3    51   B0   08C4  2304          MOVW    R1,IRP$L_MEDIA+4(R3)            ; Save user P2 buffer length
                                 30   13   08C8  2305          BEQL    60$                             ; Br if no P2 buffer present
                                           08CA  2306
                  3E A3    01   B0   08CA  2307          MOVW    S^#SS$_NORMAL,IRP$L_MEDIA+6(R3) ; Assume success
                  32 A3    18   B0   08CE  2308          MOVW    #TRIB_PRM_BUFSIZ,IRP$W_BCNT(R3) ; Set size of required buffer
                     18    51   B1   08D2  2309          CMPW    R1,#TRIB_PRM_BUFSIZ             ; If GEQU then user buffer is
                                 0A   1E   08D5  2310          BGEQU   50$                             ; large enough
          3E A3    0601 8F    B0   08D7  2311          MOVW    #SS$_BUFFEROVF,IRP$L_MEDIA+6(R3) ; Return partial success
                32 A3    51   B0   08DD  2312          MOVW    R1,IRP$W_BCNT(R3)               ; Set size to use in move
             51    F813 CF    9E   08E1  2313  50$:     MOVAB   TRIB_PARAM,R1                   ; Get address of return table
             54    0134 C5    9E   08E6  2314          MOVAB   UCB$Z_XG_DDCMP(R5),R4           ; Get trib param block address
                      1C04    30   08EB  2315          BSBW    RETURN_P2                       ; Return the P2 parameters
                50    32 A3    B0   08EE  2316          MOVW    IRP$W_BCNT(R3),R0               ; Return size of buffer
             50    50    10   78   08F2  2317          ASHL    #16,R0,R0                       ; Shift size of buffer return
                50    3E A3    B0   08F6  2318          MOVW    IRP$L_MEDIA+6(R3),R0            ; Set success of return
                                           08FA  2319
                52    38 A3    D0   08FA  2320  60$:     MOVL    IRP$L_MEDIA(R3),R2              ; Retrieve P1 buffer address
                                 04   13   08FE  2321          BEQL    70$                             ; Br if none
                62    40 A5    7D   0900  2322          MOVQ    UCB$B_DEVCLASS(R5),(R2)         ; Else, return characteristics
                51    44 A5    D0   0904  2323  70$:     MOVL    UCB$L_DEVDEPEND(R5),R1          ; Get device dependend info
```

```
                                        H 6
00000000'GF   17  0908  2324          JMP     G^EXE$FINISHIO                  ; Complete the I/O request
```

```
                                                090E  2326          .SBTTL  SENSEMODE_CTRL,  Perform SENSEMODE FDT processing for controller
                                                090E  2327
                                                090E  2328 ;++
                                                090E  2329 ; SENSEMODE_CTRL - Perform SENSEMODE FDT processing for controller
                                                090E  2330 ;
                                                090E  2331 ; Functional description:
                                                090E  2332 ;
                                                090E  2333 ; This routine performs all FDT checking for a controller SENSEMODE request.
                                                090E  2334 ; The P2 buffer if present is check for write access and if okay, a system
                                                090E  2335 ; buffer is allocated for temporarily saving the needed information. The
                                                090E  2336 ; P1 sensemode information is returned through IRP$L_MEDIA and IRP$L_MEDIA+4.
                                                090E  2337 ;
                                                090E  2338 ; Inputs:
                                                090E  2339 ;
                                                090E  2340 ;       R3 = IRP address
                                                090E  2341 ;       R4 = PCB address
                                                090E  2342 ;       R5 = UCB address
                                                090E  2343 ;       R7 = Function code
                                                090E  2344 ;
                                                090E  2345 ; Outputs:
                                                090E  2346 ;
                                                090E  2347 ;       R0 = status return for request
                                                090E  2348 ;
                                                090E  2349 ;       R3-R5 are preserved.
                                                090E  2350 ;
                                                090E  2351 ;--
                                                090E  2352 SENSEMODE_CTRL:                              ; Process controller sensemode FDT
      03 57   07   E1                           090E  2353          BBC     #IO$V_RD_MODEM,R7,2$      ; If BC not a read modem IO
            0138   31                           0912  2354          BRW     SENSE_MODEM               ; Else branch to read modem FDT
   57   0500 8F   B3                           0915  2355 2$:       BITW    #<IO$M_RD_COUNT!-         ; Check to see if either bit is
                                                091A  2356                  IO$M_CLR_COUNT>,R7        ; Is set
            03   12                           091A  2357          BNEQ    5$                        ; If NEQ then not read parameter
          0096   31                           091C  2358          BRW     40$                       ; else read parameters
   50   0167 C5   9A                           091F  2359 5$:      MOVZBL  UCB$B_XG_PROTYPE(R5),R0   ; Get protocol type
                                                0924  2360          CASE    R0,TYPE=B,<-              ; Case on protocol type
                                                0924  2361                  35$,-                     ; LAPB
                                                0924  2362                  37$,-                     ; BISYNC
                                                0924  2363                  >                         ; Fall thru on DDCMP
            50   D4                           092C  2364          CLRL    R0                        ; Assume clear count
      51    01   3C                           092E  2365          MOVZWL  S^#SS$_NORMAL,R1          ; Assume success
      00D0 C5   D5                           0931  2366          TSTL    UCB$A_XG_PRO_BUFFER(R5)   ; Check for addr of protocol buffer
            09   12                           0935  2367          BNEQ    8$                        ; If NEQ then buffer exists
   32 A3    00   B0                           0937  2368          MOVW    #0,IRP$W_BCNT(R3)         ; Else return no errors read
            50   D4                           093B  2369          CLRL    R0                        ; And complete the request
          0057   31                           093D  2370          BRW     20$
   3B 57    08   E1                           0940  2371 8$:       BBC     #IO$V_RD_COUNT,R7,15$     ; Br if not read counters
                                                0944  2372 ;
                                                0944  2373 ; Read controller counters - modifier RD_COUNT for DDCMP mode only
                                                0944  2374 ;
          019C   30                           0944  2375          BSBW    CHECK_P2                  ; Check P2 buffer
      50    14   3C                           0947  2376          MOVZWL  #SS$_BADPARAM,R0          ; Assume zero length buffer
   3C A3    51   B0                           094A  2377          MOVW    R1,IRP$L_MEDIA+4(R3)      ; Save user size of P2 buffer
            58   13                           094E  2378          BEQL    30$                       ; Br if none
      51    01   3C                           0950  2379          MOVZWL  S^#SS$_NORMAL,R1          ; Assume success
            54   DD                           0953  2380          PUSHL   R4
   54   00D0 C5   D0                           0955  2381          MOVL    UCB$A_XG_PRO_BUFFER(R5),R4 ; Set address of protocol buffer
   50   01D3 C4   3C                           095A  2382          MOVZWL  GF$W_GEBTF$R_LENGTH(R4),R0 ; Get size of buffer needed
```

```
                                                                J 6
                50    3C A3    B1  095F  2383          CMPW    IRP$L_MEDIA+4(R3),R0             ; IF GEQU then buffer ok
                         09    1E  0963  2384          BGEQU   10$
               51    0601 8F    3C  0965  2385          MOVZWL  #SS$_BUFFEROVF,R1                ; Return partial success
                50    3C A3    3C  096A  2386          MOVZWL  IRP$L_MEDIA+4(R3),R0             ; Set size of copy
                         3B    BB  096E  2387 10$:     PUSHR   #^M<R0,R1,R3,R4,R5>              ; Save the registers
            62  01E7 C4    50    28  0970  2388          MOVC3   R0,GF$K_ERRSRT+FF$K_LENGTH(R4),(R2) ; Get the errors
                         3B    BA  0976  2389          POPR    #^M<R0,R1,R3,R4,R5>
                         54 8ED0   0978  2390          POPL    R4                               ; Restore R4
                  18 57    0A    E1  097B  2391          BBC     #IO$V_CLR_COUNT,R7,20$           ; Br if not clear counts
                     57    05    3C  097F  2392 15$:     MOVZWL  #<DLK$M_GLOB!DLK$M_CLEAR>,R7     ; Set counters to read/clear
                                   0982  2393          SETIPL  UCB$B_FIPL(R5)                   ; Set to fork IPL
                         3B    BB  0986  2394          PUSHR   #^M<R0,R1,R3,R4,R5>              ; Save registers
                         58    D4  0988  2395          CLRL    R8                               ; Set no parameters to return
               55    00D0 C5    D0  098A  2396          MOVL    UCB$A_XG_PRO_BUFFER(R5),R5       ; Set address of protocol buffer
                     56    03    9A  098F  2397          MOVZBL  #DLK$C_REQEBA,R6                 ; Set operation to perform
                        F66B'    30  0992  2398          BSBW    DDCMP                            ; Branch to protocol
                         3B    BA  0995  2399          POPR    #^M<R0,R1,R3,R4,R5>
               50    50    10    78  0997  2400 20$:     ASHL    #16,R0,R0                        ; Shift size of buffer return
                     50    51    B0  099B  2401          MOVW    R1,R0                            ; Set success status
                  51    44 A5    D0  099E  2402          MOVL    UCB$L_DEVDEPEND(R5),R1           ; Set devdepend info
                00000000'GF    17  09A2  2403          JMP     G^EXE$FINISHIO                   ; Complete the request
                                   09A8  2404
                00000000'GF    17  09A8  2405 30$:     JMP     G^EXE$ABORTIO                    ; Abort the I/O request
                                   09AE  2406
                         5A    11  09AE  2407 35$:     BRB     80$
                                   09B0  2408
                     50    01    3C  09B0  2409 37$:     MOVZWL  S^#SS$_NORMAL,R0                 ; SET success and no data returned
                         4B    11  09B3  2410          BRB     70$                              ; Exit from sensemode
                                   09B5  2411 ;
                                   09B5  2412 ; Read controller parameters
                                   09B5  2413 ;
                  51    08    3C  09B5  2414 40$:     MOVZWL  S^#8,R1                          ; Size of P1 buffer if present
                     0126    30  09B8  2415          BSBW    CHECK_BUFS                       ; Check P1 and P2 buffers
                     50    01    3C  09BB  2416          MOVZWL  S^#SS$_NORMAL,R0                 ; Assume success
                   3C A3    51    D0  09BE  2417          MOVL    R1,IRP$L_MEDIA+4(R3)             ; Save user P2 buffer length
                         32    13  09C2  2418          BEQL    60$                              ; Br if no P2 buffer
                                   09C4  2419
                   3E A3    01    B0  09C4  2420          MOVW    S^#SS$_NORMAL,IRP$L_MEDIA+6(R3) ; Assume success
            32 A3    005A 8F    B0  09C8  2421          MOVW    #LINE_PRM_BUFSIZ,IRP$W_BCNT(R3) ; Set size of required buffer
               005A 8F    51    B1  09CE  2422          CMPW    R1,#LINE_PRM_BUFSIZ              ; If GEQU then user buffer is
                         0A    1E  09D3  2423          BGEQU   50$                              ; large enough
            3E A3    0601 8F    B0  09D5  2424          MOVW    #SS$_BUFFEROVF,IRP$L_MEDIA+6(R3) ; Return partial success
                  32 A3    51    B0  09DB  2425          MOVW    R1,IRP$W_BCNT(R3)                ; Set size to use in move
               51    F6A1 CF    9E  09DF  2426 50$:     MOVAB   LINE_PARAM,R1                    ; Get address of return table
                     54    55    D0  09E4  2427          MOVL    R5,R4                            ; Get line param block address
                        1B08    30  09E7  2428          BSBW    RETURN_P2                        ; Return the P2 parameters
                  50    32 A3    B0  09EA  2429 55$:     MOVW    IRP$W_BCNT(R3),R0                ; Return size of buffer
               50    50    10    78  09EE  2430          ASHL    #16,R0,R0                        ; Shift size of buffer return
                  50    3E A3    B0  09F2  2431          MOVW    IRP$L_MEDIA+6(R3),R0             ; Set size of return
                                   09F6  2432
                  52    38 A3    D0  09F6  2433 60$:     MOVL    IRP$L_MEDIA(R3),R2               ; Retrieve P1 buffer address
                         04    13  09FA  2434          BEQL    70$                              ; Br if none
                  62    40 A5    7D  09FC  2435          MOVQ    UCB$B_DEVCLASS(R5),(R2)          ; Else, return characteristics
                  51    44 A5    D0  0A00  2436 70$:     MOVL    UCB$L_DEVDEPEND(R5),R1           ; Get device dependend info
                00000000'GF    17  0A04  2437          JMP     G^EXE$FINISHIO                   ; Complete the I/O request
                                   0A0A  2438
                         00D6    30  0A0A  2439 80$:     BSBW    CHECK_P2                         ; Check P2 buffer
```

```
                38 A3   D4   0A0D  2440          CLRL    IRP$L_MEDIA(R3)                         ; Clear PSI never passes P1
              0050 8F   BB   0A10  2441          PUSHR   #^M<R4,R6>
        3E    A3   01   B0   0A14  2442          MOVW    S^#SS$_NORMAL,IRP$L_MEDIA+6(R3) ; Assume success
     56   12   17   C3   0A18  2443          SUBL3   #LAPB$K_ERREND,#LAPB$K_ERRSTRT,R6 ; Set size of LAPB error buffr
     54   00D0 C5   D0   0A1C  2444          MOVL    UCB$A_XG_PRO_BUFFER(R5),R4      ; Set address of protocol buffer
                06   12   0A21  2445          BNEQ    85$                             ; If NEQ then buffer exists
          32 A3   00   B0   0A23  2446          MOVW    #0,IRP$W_BCNT(R3)               ; Else return no errors read
                C1   11   0A27  2447          BRB     55$
          56    51   B1   0A29  2448 85$:     CMPW    R1,R6                           ; IF gequ then buff is large
                09   1E   0A2C  2449          BGEQU   90$                             ; encugh
  3E A3   0601  8F   B0   0A2E  2450          MOVW    #SS$_BUFFEROVF,IRP$L_MEDIA+6(R3) ; Set error
          56    51   B0   0A34  2451          MOVW    R1,R6
          32 A3   56   B0   0A37  2452 90$:     MOVW    R6,IRP$W_BCNT(R3)               ; Set size of transfer
                38   BB   0A3B  2453          PUSHR   #^M<R3,R4,R5>
  62    12 A4   56   28   0A3D  2454          MOVC3   R6,LAPB$K_ERRSTRT(R4),(R2)      ; Copy counters to user buffer
                38   BA   0A42  2455          POPR    #^M<R3,R4,R5>
             16 A4   94   0A44  2456          CLRB    LAPB$B_DEI(R4)                  ; Clear the counter
              0050 8F   BA   0A47  2457          POPR    #^M<R4,R6>
                9D   11   0A4B  2458          BRB     55$                             ; Jump to complete the req
```

```
                        0A4D  2460          .SBTTL  SENSE_MODEM - Perform SENSEMODE READ_MODEM FDT processing
                        0A4D  2461 ;++
                        0A4D  2462 ;SENSE_MODEM - Perform SENSEMODE READ_MODEM FDT processing
                        0A4D  2463 ;
                        0A4D  2464 ; This routine performs all FDT checking for a SENSEMODE read modem reqeust.
                        0A4D  2465 ; The P1 buffer is checked for write access and the request is queued to
                        0A4D  2466 ; the driver.
                        0A4D  2467 ;
                        0A4D  2468 ;       Inputs:
                        0A4D  2469 ;               R3 = IRP address
                        0A4D  2470 ;               R4 = PCB address
                        0A4D  2471 ;               R5 = UCB address
                        0A4D  2472 ;               R7 = Fucntion code
                        0A4D  2473 ;
                        0A4D  2474 ;       Outputs
                        0A4D  2475 ;               R0 = Status for return
                        0A4D  2476 ;               R3-R5 are preserved
                        0A4D  2477 ;--
                        0A4D  2478 SENSE_MODEM:
          51    04   3C  0A4D  2479          MOVZWL  S^#4,R1                         ; Size of P1 buffer
             00BA   30  0A50  2480          BSBW    CHECK_P1                        ; Check access to P1
                        0A53  2481                                                  ; (no return on no access)
             00CA   30  0A53  2482          BSBW    ALLOC_P2BUF                     ; Allocate a P2 buffer
            03 50   E8  0A56  2483          BLBS    R0,10$                          ; Branch if success
             FABE   31  0A59  2484          BRW     ABORTIO                         ; Else abort the request
                        0A5C  2485
       51    2C A3   D0  0A5C  2486 10$:    MOVL    IRP$L_SVAPTE(R3),R1             ; Get system buffer address
 04 A1    38 A3   D0  0A60  2487          MOVL    IRP$L_MEDIA(R3),P2B_L_BUFFER(R1) ; Set user buffer VA
          38 A3   D4  0A65  2488          CLRL    IRP$L_MEDIA(R3)                 ; Clear buffer
    21 A3    05   90  0A68  2489          MOVB    #XG_FC_V_READ_MODEM,IRP$B_XGFUNC(R3) ; Set funcition
                        0A6C  2490          ;BRB    QUEPKT                          ;Give request o driver
                        0A6C  2491
                        0A6C  2492
                        0A6C  2493 ;
                        0A6C  2494 ; I/O request packet to driver
                        0A6C  2495 ;
                        0A6C  2496 QUEPKT:
                        0A6C  2497          SETIPL  UCB$B_FIPL(R5)                  ; Queue packet to driver
00000000'GF    16   0A70  2498          JSB     G^IOC$INITIATE                  ; Initiate I/O reuqest
00000000'GF    17   0A76  2499          JMP     G^EXE$QIORETURN                 ; Lower IPL, and return
                        0A7C  2500
```

```
                     0A7C  2502          .SBTTL  GET_CHAR_BUFS,  Get P1 and P2 characteristics buffers
                     0A7C  2503
                     0A7C  2504 ;++
                     0A7C  2505 ; GET_CHAR_BUFS - Get P1 and P2 characteristics buffers
                     0A7C  2506 ;
                     0A7C  2507 ; Functional description:
                     0A7C  2508 ;
                     0A7C  2509 ; This routine saves the P1 and P2 buffers for later use by the driver.
                     0A7C  2510 ; The P1 buffer is saved in the IRP (IRP$L_MEDIA) as a quadword value.
                     0A7C  2511 ; The P2 buffer is saved by allocating the appropriate amount of memory from
                     0A7C  2512 ; non-paged pool.   The user's quota is checked before the allocation is made.
                     0A7C  2513 ; And the non-paged pool buffer is charged against the user's quota. The P2
                     0A7C  2514 ; system buffer address is passed in IRP$L_SVAPTE of the IRP.
                     0A7C  2515 ;
                     0A7C  2516 ;
                     0A7C  2517 ; Inputs:
                     0A7C  2518 ;
                     0A7C  2519 ;       R3 = IRP address
                     0A7C  2520 ;       R4 = PCB address
                     0A7C  2521 ;       R5 = UCB address
                     0A7C  2522 ;
                     0A7C  2523 ; Outputs:
                     0A7C  2524 ;
                     0A7C  2525 ;       R0 = status of buffers
                     0A7C  2526 ;
                     0A7C  2527 ;       R3-R5 are preserved.
                     0A7C  2528 ;
                     0A7C  2529 ;--
                     0A7C  2530
                     0A7C  2531 GET_CHAR_BUFS:                                   ; Get characteristics buffers
                     0A7C  2532 ;
                     0A7C  2533 ; Check access to P1 buffer and save P1 characteristics
                     0A7C  2534 ;
        38 A3   7C   0A7C  2535         CLRQ    IRP$L_MEDIA(R3)                 ; Reset P1 chars
     52    6C   D0   0A7F  2536         MOVL    P1(AP),R2                       ; Get address of P1 char buf
           11   13   0A82  2537         BEQL    10$                             ; Branch if no P1 buffer
     50    0C   3C   0A84  2538         MOVZWL  #SS$_ACCVIO,R0                  ; Assume access violation
                     0A87  2539         IFNORD  #8,(R2),20$                     ; Check access
  38 A3    62   7D   0A8D  2540         MOVQ    (R2),IRP$L_MEDIA(R3)            ; Save P1 characteristics
  38 A3    01   90   0A91  2541         MOVB    #1,IRP$L_MEDIA(R3)              ; Indicate valid P1 buffer
                     0A95  2542 ;
                     0A95  2543 ; Check access to P2 buffer and check process's buffer quota
                     0A95  2544 ;
  52    04 AC   D0   0A95  2545 10$:    MOVL    P2(AP),R2                       ; Get address P2 char buf desc
           42   13   0A99  2546         BEQL    40$                             ; Br if no P2 buffer
     50    0C   3C   0A9B  2547         MOVZWL  #SS$_ACCVIO,R0                  ; Assume access violation
                     0A9E  2548         IFNORD  #8,(R2),20$                     ; Check access to descriptor
     51    62   3C   0AA4  2549         MOVZWL  (R2),R1                         ; Get buffer length in bytes
     51    01   CA   0AA7  2550         BICL    #1,R1                           ; Must be multiple of 2 bytes
           31   13   0AAA  2551         BEQL    40$                             ; Br if size is zero
  52    04 A2   D0   0AAC  2552         MOVL    DSC$A_POINTER(R2),R2            ; Get buffer address
     50    52   D0   0AB0  2553         MOVL    R2,R0                           ; Copy buffer address
           0C   BB   0AB3  2554         PUSHR   #^M<R2,R3>                      ; Save R2, R3
00000000'GF    16   0AB5  2555         JSB     G^EXE$WRITECHKR                 ; Check entire buffer
        06 50   E9   0ABB  2556         BLBC    R0,15$                          ; Branch if error
00000000'GF    16   0ABE  2557         JSB     G^EXE$BUFQUOPRC                 ; Check for buffered quota
           0C   BA   0AC4  2558 15$:    POPR    #^M<R2,R3>                      ; Restore R2, R3
```

```
                01 50    E8  0AC6  2559          BLBS    R0,30$                          ; Branch if quota ok
                         05  0AC9  2560 20$:     RSB                                     ; Return
                             0ACA  2561
                             0ACA  2562 ;
                             0ACA  2563 ; Quota OKAY, allocate buffer and copy info.
                             0ACA  2564 ;
                  0053   30  0ACA  2565 30$:     BSBW    ALLOC_P2BUF                     ; Allocate buffer
                10 50    E9  0ACD  2566          BLBC    R0,50$                          ; Br if error
          50    2C A3    D0  0AD0  2567          MOVL    @RP$L_SVAPTE(R3),R0             ; Get P2 buffer address
                   38    BB  0AD4  2568          PUSHR   #^M<R3,R4,R5>                   ; Save sacred registers
       0C A0  62   51    28  0AD6  2569          MOVC3   R1,(R2),P2B_T_DATA(R0)          ; Save P2 char buffer
                   38    BA  0ADB  2570          POPR    #^M<R3,R4,R5>                   ; Restore registers
             50    01    3C  0ADD  2571 40$:     MOVZWL  S^#SS$_NORMAL,R0                ; Set success
                         05  0AE0  2572 50$:     RSB                                     ; Return
```

```
                     0AE1  2574          .SBTTL  CHECK_BUFS,  Check P1 and P2 buffers for write access
                     0AE1  2575
                     0AE1  2576 ;++
                     0AE1  2577 ; CHECK_BUFS - Check P1 and P2 buffers for write access
                     0AE1  2578 ;
                     0AE1  2579 ; Functional description:
                     0AE1  2580 ;
                     0AE1  2581 ; This routines checks the P1 and P2 buffers for write access if supplied.
                     0AE1  2582 ;
                     0AE1  2583 ; Inputs:
                     0AE1  2584 ;
                     0AE1  2585 ;       R1 = Size of P1 buffer needed for write access
                     0AE1  2586 ;       R3 = IRP address
                     0AE1  2587 ;       R4 = PCB address
                     0AE1  2588 ;       R5 = UCB address
                     0AE1  2589 ;       R7 = Function code
                     0AE1  2590 ;       R9 = CDB address
                     0AE1  2591 ;
                     0AE1  2592 ; Outputs:
                     0AE1  2593 ;
                     0AE1  2594 ;       R1 = Length of P2 buffer (zero if no P2 buffer)
                     0AE1  2595 ;       R2 = Address of P2 buffer in user's process space
                     0AE1  2596 ;
                     0AE1  2597 ;       R0-R2 are destroyed.
                     0AE1  2598 ;
                     0AE1  2599 ;       No RETURN on NO ACCESS
                     0AE1  2600 ;
                     0AE1  2601 ; Implicit Outputs:
                     0AE1  2602 ;
                     0AE1  2603 ;       IRP$V_FUNC bit set in IRP$W_STS by EXE$READCHK subroutine.
                     0AE1  2604 ;
                     0AE1  2605 ;--
                     0AE1  2606
                     0AE1  2607 CHECK_BUFS:
           2A    10  0AE1  2608          BSBB    CHECK_P1                        ; Check P1 buffer
                     0AE3  2609 CHECK_P2:
           51    D4  0AE3  2610          CLRL    R1                              ; Assume no P2 buffer desc
      52   04  AC    D0  0AE5  2611          MOVL    P2(AP),R2                       ; Get address of P2 desc
           1B    13  0AE9  2612          BEQL    10$                             ; Br if no P2
                     0AEB  2613          IFNORD  #8,(R2),ACCESS                  ; Br if no access
      51   62    3C  0AF1  2614          MOVZWL  (R2),R1                         ; Get length of buffer
      51   01    CA  0AF4  2615          BICL    #1,R1                           ; Must be multiple of 2 bytes
           0D    13  0AF7  2616          BEQL    10$                             ; Br if zero
      50   04  A2    D0  0AF9  2617          MOVL    DSC$A_POINTER(R2),R0            ; Get buffer address
00000000'GF    16  0AFD  2618          JSB     G^EXE$READCHK                   ; Check write access to buffer
                     0B03  2619                                                  ; (no return no access)
                     0B03  2620                                                  ; Also sets IRP$V_FUNC in IRP
      52   50    D0  0B03  2621          MOVL    R0,R2                           ; Copy buffer address
                05  0B06  2622 10$:     RSB                                     ; Return to caller
                     0B07  2623
      50   0C    3C  0B07  2624 ACCESS: MOVZWL  #SS$_ACCVIO,R0                  ; Return access violation
           FA0D  31  0B0A  2625          BRW     ABORTIO                         ; Abort the I/O request
```

```
                     0B0D  2627          .SBTTL  CHECK_P1,  Check P1 buffer address for write access
                     0B0D  2628
                     0B0D  2629 ;++
                     0B0D  2630 ; CHECK_P1 - Check P1 buffer address for write access
                     0B0D  2631 ;
                     0B0D  2632 ; Functional description:
                     0B0D  2633 ;
                     0B0D  2634 ;  This routine checks the P1 buffer and if okay, the buffer address
                     0B0D  2635 ;  is saved in IRP$L_MEDIA of the IRP.
                     0B0D  2636 ;
                     0B0D  2637 ; Inputs:
                     0B0D  2638 ;
                     0B0D  2639 ;       R1 = Size of buffer for write access
                     0B0D  2640 ;       R3 = IRP address
                     0B0D  2641 ;       R4 = PCB address
                     0B0D  2642 ;       R5 = UCB address
                     0B0D  2643 ;       R7 = Function code
                     0B0D  2644 ;       R9 = CDB address
                     0B0D  2645 ;
                     0B0D  2646 ; Outputs:
                     0B0D  2647 ;
                     0B0D  2648 ;       R0 is destroyed.
                     0B0D  2649 ;
                     0B0D  2650 ;       No RETURN on NO ACCESS.
                     0B0D  2651 ;
                     0B0D  2652 ; Implicit Outputs:
                     0B0D  2653 ;
                     0B0D  2654 ;       IRP$L_MEDIA(R3) = User P1 buffer address.
                     0B0D  2655 ;       IRP$V_FUNC bit set in IRP$W_STS by EXE$READCHK subroutine.
                     0B0D  2656 ;
                     0B0D  2657 ;--
                     0B0D  2658
                     0B0D  2659 CHECK_P1:
        38 A3   D4   0B0D  2660          CLRL    IRP$L_MEDIA(R3)             ; Assume no P1 buffer
     50    6C   D0   0B10  2661          MOVL    P1(AP),R0                   ; Get address of user buffer
           0A   13   0B13  2662          BEQL    10$                         ; Br if none
00000000'GF    16   0B15  2663          JSB     G^EXE$READCHK               ; Check access to buffer
                    0B1B  2664                                              ; (No return - no access)
 38 A3    50   D0   0B1B  2665          MOVL    R0,IRP$L_MEDIA(R3)          ; Save P1 buffer address in IRP
                    05   0B1F  2666 10$:    RSB                             ; Return to caller
```

```
                     0B20  2668          .SBTTL  ALLOC_P2BUF,  Allocate a P2 buffer and charge user's quota
                     0B20  2669
                     0B20  2670 ;++
                     0B20  2671 ; ALLOC_P2BUF - Allocate a P2 buffer and charge user's quota
                     0B20  2672 ;
                     0B20  2673 ; Functional description:
                     0B20  2674 ;
                     0B20  2675 ; This routine allocates a system buffer ard returns the address in the IRP at
                     0B20  2676 ; IRP$L_SVAPTE.  The size of the allocation, including buffer header must
                     0B20  2677 ; be at least 24 bytes in length.
                     0B20  2678 ;
                     0B20  2679 ; Inputs:
                     0B20  2680 ;
                     0B20  2681 ;       R1 = Size of allocation desired
                     0B20  2682 ;       R3 = IRP address
                     0B20  2683 ;
                     0B20  2684 ; Outputs:
                     0B20  2685 ;
                     0B20  2686 ;       R0 = status of request
                     0B20  2687 ;
                     0B20  2688 ;       R1-R5 are preserved.
                     0B20  2689 ;
                     0B20  2690 ; Implicit Outputs:
                     0B20  2691 ;
                     0B20  2692 ;       IRP$L_SVAPTE(R3) = address of system buffer
                     0B20  2693 ;       IRP$W_BOFF(R3) = byte count charged to user's process
                     0B20  2694 ;       IRP$W_BCNT(R3) = original byte count requested
                     0B20  2695 ;
                     0B20  2696 ;       All parts of the P2 buffer header are initialized, except for the
                     0B20  2697 ;       user's P2 buffer address.
                     0B20  2698 ;
                     0B20  2699 ;--
                     0B20  2700
                     0B20  2701 ALLOC_P2BUF:                                    ; Allocate a non-paged buffer
          51    D5   0B20  2702          TSTL    R1                             ; Zero length buffer?
          50    13   0B22  2703          BEQL    30$                            ; Br if yes
          0E    BB   0B24  2704          PUSHR   #^M<R1,R2,R3>                  ; Save registers
 32 A3    51    B0   0B26  2705          MOVW    R1,IRP$W_BCNT(R3)              ; Save original byte count
    0C    51    D1   0B2A  2706          CMPL    R1,S^#24-P2B_C_LENGTH          ; Is buffer big enough?
          03    1A   0B2D  2707          BGTRU   5$                             ; Br if yes
    51    0C    D0   0B2F  2708          MOVL    S^#24-P2B_C_LENGTH,R1          ; Else, set size to minimum
    51    0C    C0   0B32  2709 5$:      ADDL2   S^#P2B_C_LENGTH,R1             ; Add in size of header
00000000'GF     16   0B35  2710          JSB     G^EXE$BUFQUOPRC                ; Check for buffered quota
      0D 50     E9   0B3B  2711          BLBC    R0,10$                         ; Branch if quota bad
                     0B3E  2712 ;
                     0B3E  2713 ; Quota OKAY, allocate buffer and copy info.
                     0B3E  2714 ;
          51    DD   0B3E  2715          PUSHL   R1                             ; Save size to charge user
00000000'GF     16   0B40  2716          JSB     G^EXE$ALONONPAGED              ; Go allocate a buffer
      05 50     E8   0B46  2717          BLBS    R0,20$                         ; Br if success
          8E    D5   0B49  2718          TSTL    (SP)+                          ; Pop saved size
          0E    BA   0B4B  2719 10$:     POPR    #^M<R1,R2,R3>                  ; Restore registers
                05   0B4D  2720          RSB                                    ; Return with error code in R0
                     0B4E  2721 ;
                     0B4E  2722 ; System buffer allocated decrement user's quota
                     0B4E  2723 ;
          53 8ED0    0B4E  2724 20$:     POPL    R3                             ; Restore user quota charge
```

```
                62    0C A2   9E  0B51  2725          MOVAB   P2B_T_DATA(R2),P2B_L_POINTER(R2) ; Set address to start of data
                08 A2    53   B0  0B55  2726          MOVW    R3,P2B_W_SIZE(R2)                ; Save buffer size in buffer
                0A A2    13   90  0B59  2727          MOVB    S^#DYN$C_BUFIO,P2B_B_TYPE(R2)    ; Set structure type
                   50    52   D0  0B5D  2728          MOVL    R2,R0                            ; Save P2 char buf addr
             52    0080 C4   D0  0B60  2729          MOVL    PCB$L_JIB(R4),R2                 ; Get JIB address
                20 A2    53   C2  0B65  2730          SUBL    R3,JIB$L_BYTCNT(R2)              ; Decrement user's quota
                   0E    BA   0B69  2731          POPR    #^M<R1,R2,R3>                    ; Restore registers
                2C A3    50   D0  0B6B  2732          MOVL    R0,IRP$L_SVAPTE(R3)              ; Save P2 buffer address in IRP
             30 A3    08 A0   B0  0B6F  2733          MOVW    P2B_W_SIZE(R0),IRP$W_BOFF(R3)    ; Return buffer size in IRP
                   50    01   3C  0B74  2734 30$:     MOVZWL  S^#SS$_NORMAL,R0                 ; Set success
                               05  0B77  2735          RSB                                      ; Return to caller
```

```
                                0B78  2737          .SBTTL  CHG_UCB,  Change UCB parameter values
                                0B78  2738
                                0B78  2739 ;++
                                0B78  2740 ; CHG_UCB - Change UCB parameter values
                                0B78  2741 ;
                                0B78  2742 ; Functional description:
                                0B78  2743 ;
                                0B78  2744 ; This routine is called to initialize the UCB with new P1 and P2 buffer
                                0B78  2745 ; characteristics. It is assumed here that the parameters have already
                                0B78  2746 ; been validated.
                                0B78  2747 ;
                                0B78  2748 ; Inputs:
                                0B78  2749 ;
                                0B78  2750 ;       R3 = IRP address
                                0B78  2751 ;       R5 = UCB address
                                0B78  2752 ;       R6 = Protocol mode in which to run the driver
                                0B78  2753 ;
                                0B78  2754 ;       IPL = FIPL
                                0B78  2755 ;
                                0B78  2756 ; Outputs:
                                0B78  2757 ;
                                0B78  2758 ;       R0-R2 = destroyed.
                                0B78  2759 ;
                                0B78  2760 ;--
                                0B78  2761
                                0B78  2762 CHG_UCB:                                     ; Change UCB parameters
                          54    DD  0B78  2763          PUSHL   R4                      ; Save R4
                                0B7A  2764 ;
                                0B7A  2765 ; Call to set the start up default characteristics for a given protocol
                                0B7A  2766 ;
                          0086  30  0B7A  2767          BSBW    SET_DEFAULT_CHAR
                                0B7D  2768
                       44 38 A3  E9  0B7D  2769          BLBC    IRP$L_MEDIA(R3),10$     ; Br if no P1 buffer
                                0B81  2770 ;
                                0B81  2771 ; Set new P1 buffer characteristics. We will use the P1 buffer if no
                                0B81  2772 ; protocol or the DDCMP protocol was specified in the P2 buffer.
                                0B81  2773 ; There is no interface to set LAPB or BISYNC via the P1 buffer.
                                0B81  2774 ;
                    0167 C5   02  91  0B81  2775          CMPB    #XG$C_PROTYPE_DDCMP,UCB$B_XG_PROTYPE(R5) ; If not DDCMP then can
                             3D   12  0B86  2776          BNEQ    10$                     ; not use P1 buffer
                             02   E0  0B88  2777          BBS     #XG_DS_V_INITED,-       ; Br if device inited
                          05 68 A5        0B8A  2778                  UCB$W_DEVSTS(R5),5$
                             3A A3   B0  0B8D  2779          MOVW    IRP$L_MEDIA+2(R3),-     ; Set new buffer size
                             42 A5       0B90  2780                  UCB$W_DEVBUFSIZ(R5)
                 44 A5   18   08  00   F0  0B92  2781 5$:     INSV    #0,#8,#24,UCB$L_DEVDEPEND(R5) ; Reset all Read/Write flags
                             3C A3   C8  0B98  2782          BISL    IRP$L_MEDIA+4(R3),-     ; Set new characteristics
                             44 A5       0B9B  2783                  UCB$L_DEVDEPEND(R5)
                                0B9D  2784 ;
                                0B9D  2785 ; Now update UCB based on P1 buffer
                                0B9D  2786 ;
                                0B9D  2787          ASSUME  NMA$C_LINPR_POI EQ 0
                                0B9D  2788
                          0150 C5   94  0B9D  2789          CLRB    UCB$B_XG_PRO(R5)        ; Assume point to point mode
                       05 44 A5   07  E1  0BA1  2790          BBC     #XM$V_CHR_TRIB,UCB$L_DEVDEPEND(R5),6$ ; Branch BC ddcmp point
                          0150 C5   02  90  0BA6  2791          MOVB    #NMA$C_LINPR_TRI,UCB$B_XG_PRO(R5) ; Else must be a trib station
                                0BAB  2792
                                0BAB  2793          ASSUME  NMA$C_LINCN_NOR EQ 0
```

```
                          0BAB  2794
                0152 C5   94    0BAB  2795 6$:     CLRB    UCB$B_XG_CON(R5)                    ; Assume normal mode
                     01   E1    0BAF  2796          BBC     #XM$V_CHR_LOOPB,-                   ; Br if not loopback mode
                04 44 A5          0BB1  2797                  UCB$L_DEVDEPEND(R5),7$              ;
                                  0BB4  2798          ASSUME  NMA$C_LINCN_LOO EQ 1
                0152 C5   96    0BB4  2799          INCB    UCB$B_XG_CON(R5)                    ; Must be loopback mode
                                  0BB8  2800 7$:      ASSUME  NMA$C_DPX_FUL EQ 0
                0151 C5   94    0BB8  2801          CLRB    UCB$B_XG_DUP(R5)                    ; Assume full duplex
                     02   E1    0BBC  2802          BBC     #XM$V_CHR_HDPLX,-                   ; Br if not half duplex
                04 44 A5          0BBE  2803                  UCB$L_DEVDEPEND(R5),10$             ;
                                  0BC1  2804          ASSUME  NMA$C_DPX_HAL EQ 1
                0151 C5   96    0BC1  2805          INCB    UCB$B_XG_DUP(R5)                    ; Must be half duplex
                                  0BC5  2806
                                  0BC5  2807 ;
                                  0BC5  2808 ; Set new P2 buffer characteristics
                                  0BC5  2809 ;
                     54   55    D0    0BC5  2810 10$:     MOVL    R5,R4                       ; Copy UCB address
           51    F4B8 CF    9E    0BC8  2811          MOVAB   LINE_PARAM,R1               ; Get address of verification table
                   18B1    30    0BCD  2812          BSBW    UPDATE_P2                   ; Update the UCB
                                  0BD0  2813 ;
                                  0BD0  2814 ; Set device characteristics and device mode definition
                                  0BD0  2815 ;
                   44 A5    94    0BD0  2816          CLRB    UCB$L_DEVDEPEND(R5)                 ; Reset all characteristics
                                  0BD3  2817
                                  0BD3  2818          ASSUME  NMA$C_LINPR_POI EQ 0
                                  0BD3  2819          ASSUME  NMA$C_LINPR_TRI EQ 2
                                  0BD3  2820          ASSUME  NMA$C_LINPR_LAPB EQ 5
                                  0BD3  2821          ASSUME  NMA$C_LINPR_BSY EQ 9
                                  0BD3  2822
           52    0150 C5    9A    0BD3  2823          MOVZBL  UCB$B_XG_PRO(R5),R2         ; Get protocol mode
                     0A    13    0BD8  2824          BEQL    20$                         ; Br if point to point mode
                05   52    91    0BDA  2825          CMPB    R2,#NMA$C_LINPR_LAPB        ; If LAPB or BISYNC branch
                     05    1E    0BDD  2826          BGEQU   20$
                                  0BDF  2827          SETBIT  #XM$V_CHR_TRIB,-            ; Indicate tributary station
                                  0BDF  2828                  UCB$L_DEVDEPEND(R5)         ; If noother prot set then TRIB
                                  0BE4  2829 20$:     SETBIT  #XM$V_CHR_HDPLX,UCB$L_DEVDEPEND(R5) ; Assume half duplex mode
                                  0BE9  2830          ASSUME  NMA$C_DPX_FUL EQ 0
                0151 C5   95    0BE9  2831          TSTB    UCB$B_XG_DUP(R5)                    ; Full duplex mode?
                     05   12    0BED  2832          BNEQ    30$                         ; Br if no - okay
                                  0BEF  2833          CLRBIT  #XM$V_CHR_HDPLX,UCB$L_DEVDEPEND(R5) ; Set to full duplex mode
                                  0BF4  2834 30$:     ASSUME  NMA$C_LINCN_NOR EQ 0
                0152 C5   95    0BF4  2835          TSTB    UCB$B_XG_CON(R5)                    ; Is line in normal mode?
                     05   13    0BF8  2836          BEQL    40$                         ; Br if yes
                                  0BFA  2837          SETBIT  #XM$V_CHR_LOOPB,UCB$L_DEVDEPEND(R5) ; Set loopback mode
                     54 8ED0    0BFF  2838 40$:     POPL    R4                          ; Restore R4
                          05    0C02  2839          RSB
                                  0C03  2840
```

```
                     0C03  2842          .SBTTL  SET_DEFAULT_CHAR, Set default characteristics for given protocol
                     0C03  2843
                     0C03  2844 ;++
                     0C03  2845 ; SET_DEFAULT_CHAR - Set default characteristics for given protocol
                     0C03  2846 ;
                     0C03  2847 ; This routine sets the default start up characteristics for a given
                     0C03  2848 ; protocol supported by the driver.
                     0C03  2849 ;
                     0C03  2850 ;       INPUTS  R5 = UCB address
                     0C03  2851 ;               R6 = Protocol type
                     0C03  2852 ;
                     0C03  2853 ;       OUTPUTS All registers are preserved
                     0C03  2854 ;
                     0C03  2855 ;--
                     0C03  2856 SET_DEFAULT_CHAR:
          07   BB   0C03  2857          PUSHR   #^M<R0,R1,R2>
52   0144 C5   9E   0C05  2858          MOVAB   UCB$Z_XG_SYNC(R5),R2            ; Set address of parameters
                     0C0A  2859 ;
                     0C0A  2860 ; Assume that the protocol type will be DDCMP, thus set all the block of
                     0C0A  2861 ; defaults to point to DDCMP's.
                     0C0A  2862 ;
                     0C0A  2863
     50   08   3C   0C0A  2864          MOVZWL  #DEF_SYNC_PARAMSZ,R0            ; Set size of defaults
51   F525 CF   9E   0C0D  2865          MOVAB   DEF_SYNC_PARAM,R1               ; Set address of defaults
0167 C5   02   90   0C12  2866          MOVB    #XG$C_PROTYPE_DDCMP,UCB$B_XG_PROTYPE(R5) ; Set prot type
                     0C17  2867 ;
                     0C17  2868 ; Check to see if one of the other protocols has been specified and it so
                     0C17  2869 ; then change the pointers to point at their defaults. We must use the
                     0C17  2870 ; compares here, becaues the UCB$B_XG_PROTYPE field has not as yet been
                     0C17  2871 ; set up.
                     0C17  2872 ;
     56   05   91   0C17  2873          CMPB    #NMA$C_LINPR_LAPB,R6
          0E   13   0C1A  2874          BEQL    20$
     56   09   91   0C1C  2875          CMPB    #NMA$C_LINPR_BSY,R6
          18   13   0C1F  2876          BEQL    30$
     82   81   90   0C21  2877 10$:     MOVB    (R1)+,(R2)+                     ; Set next default
       FA 50   F5   0C24  2878          SOBGTR  R0,10$                          ; Loop on all parameters
          07   BA   0C27  2879          POPR    #^M<R0,R1,R2>
               05   0C29  2880          RSB
                     0C2A  2881 ;
                     0C2A  2882 ; Set default pointers for LAPB
                     0C2A  2883 ;
     50   08   3C   0C2A  2884 20$:     MOVZWL  #DEF_LAPB_PARAMSZ,R0            ; Set size of defaults
51   F50D CF   9E   0C2D  2885          MOVAB   DEF_LAPB_PARAM,R1               ; Set address of defaults
0167 C5   00   90   0C32  2886          MOVB    #XG$C_PROTYPE_LAPB,UCB$B_XG_PROTYPE(R5) ; Set prot type
          E8   11   0C37  2887          BRB     10$
                     0C39  2888 ;
                     0C39  2889 ; Set default pointers for BISYNC
                     0C39  2890 ;
     50   08   3C   0C39  2891 30$:     MOVZWL  #DEF_BISYNC_PARAMSZ,R0          ; Set size of defaults
51   F506 CF   9E   0C3C  2892          MOVAB   DEF_BISYNC_PARAM,R1             ; Set address of defaults
0167 C5   01   90   0C41  2893          MOVB    #XG$C_PROTYPE_BISYNC,UCB$B_XG_PROTYPE(R5) ; Set prot type
          D9   11   0C46  2894          BRB     10$
                     0C48  2895
```

```
                        0C48  2896          .SBTTL  CHG_TRIB,  Change trib parameter values
                        0C48  2897
                        0C48  2898 ;++
                        0C48  2899 ; CHG_TRIB - Change trib parameter values
                        0C48  2900 ;
                        0C48  2901 ; Functional description:
                        0C48  2902 ;
                        0C48  2903 ; This routine is called to initialize the trib parameter block with
                        0C48  2904 ; new P1 and P2 buffer characteristics. It is assumed here that the
                        0C48  2905 ; parameters have already been validated.
                        0C48  2906 ;
                        0C48  2907 ; Inputs:
                        0C48  2908 ;
                        0C48  2909 ;       R3 = IRP address
                        0C48  2910 ;       R5 = UCB address
                        0C48  2911 ;
                        0C48  2912 ;       IPL = FIPL
                        0C48  2913 ;
                        0C48  2914 ; Outputs:
                        0C48  2915 ;
                        0C48  2916 ;       R1,R2 = destroyed.
                        0C48  2917 ;--
                        0C48  2918
                        0C48  2919 CHG_TRIB:                                    ; Validate P2 buffer
              54    DD  0C48  2920          PUSHL   R4                        ; Save R4
    51  F4AA CF    9E  0C4A  2921          MOVAB   TRIB_PARAM,R1             ; Get addre of verify table
    54  0134 C5    9E  0C4F  2922          MOVAB   UCB$Z_XG_DDCMP(R5),R4     ; Get the addr of param block
        10 38 A3    E9  0C54  2923          BLBC    IRP$L_MEDIA(R3),10$       ; Br if no P1 buffer
                        0C58  2924 ;
                        0C58  2925 ; Set new P1 buffer characteristics
                        0C58  2926 ;
 44 A5    3C A3    C8  0C58  2927          BISL    IRP$L_MEDIA+4(R3),UCB$L_DEVDEPEND(R5) ; Set new char
                        0C5D  2928
                        0C5D  2929          ASSUME  NMA$C_STATE_ON EQ 0
                        0C5D  2930          ASSUME  NMA$C_STATE_OFF EQ 1
                        0C5D  2931
          08 A4    94  0C5D  2932          CLRB    DLK$B_MAINT(R4)           ; Assume MOP
 03 44 A5    00    E0  0C60  2933          BBS     #XM$V_CHR_MOP,UCB$L_DEVDEPEND(R5),10$ ; Branch BS if true
          08 A4    96  0C65  2934          INCB    DLK$B_MAINT(R4)           ; Set NORMAL mode
                        0C68  2935 ;
                        0C68  2936 ; Set new P2 buffer characteristics
                        0C68  2937 ;
            1816   30  0C68  2938 10$:     BSBW    UPDATE_P2                 ; Update trib parameters
                        0C6B  2939          CLRBIT  #XM$V_CHR_MOP,UCB$L_DEVDEPEND(R5) ; Assume NORMAL mode
       05 08 A4    E8  0C70  2940          BLBS    DLK$B_MAINT(R4),20$       ; Branch LBS if true
                        0C74  2941          SETBIT  #XM$V_CHR_MOP,UCB$L_DEVDEPEND(R5) ; Set MOP mode
            54 8ED0  0C79  2942 20$:     POPL    R4                        ; Restore R4
                 05  0C7C  2943          RSB                               ; Return to caller
                        0C7D  2944
```

```
                                                0C7D  2946          .SBTTL  STARTIO - Start setmode I/O operation
                                                0C7D  2947 ;++
                                                0C7D  2948 ; STARTIO - Start setmode operation
                                                0C7D  2949 ;
                                                0C7D  2950 ; FUNCTIONAL DESCRIPTION:
                                                0C7D  2951 ;
                                                0C7D  2952 ; This routine is entered to process a setmode request.  All setmode requests
                                                0C7D  2953 ; are queued through the UCB to single-stream them.
                                                0C7D  2954 ;
                                                0C7D  2955 ; SETMODE FUNCTIONS --
                                                0C7D  2956 ;
                                                0C7D  2957 ;       For all functions a change in the characteristics is done.
                                                0C7D  2958 ;
                                                0C7D  2959 ;       For control startup, the UCB is initialized; mapping registers for
                                                0C7D  2960 ;       receives and transmits are allocated and saved in the UCB; and the
                                                0C7D  2961 ;       DMF sync line is master reset.
                                                0C7D  2962 ;       For trib startup, the DMF sync line is loaded with its char; the
                                                0C7D  2963 ;       protocol is set up with charateristics and started; and finally
                                                0C7D  2964 ;       transmits and receives are started on the board.
                                                0C7D  2965 ;
                                                0C7D  2966 ;       For control shutdown, mapping registers are deallocated and returned;
                                                0C7D  2967 ;       all quotas are returned; and a call is made to trib shutdown to shut
                                                0C7D  2968 ;       the trib.
                                                0C7D  2969 ;       For trib shutdown, the device is master cleared; all buffers and
                                                0C7D  2970 ;       IRP's are returned; and the protocol is halted.
                                                0C7D  2971 ;
                                                0C7D  2972 ; INPUTS:
                                                0C7D  2973 ;
                                                0C7D  2974 ;       R3 = I/O packet address
                                                0C7D  2975 ;       R5 = UCB address
                                                0C7D  2976 ;
                                                0C7D  2977 ; OUTPUTS:
                                                0C7D  2978 ;
                                                0C7D  2979 ;       R3 and R5 preserved.
                                                0C7D  2980 ;--
                                                0C7D  2981          .ENABL  LSB
                                                0C7D  2982 STARTIO:                                             ; Start I/O routine
51    21 A3    9A                               0C7D  2983          MOVZBL  IRP$B_XGFUNC(R3),R1                 ; Get the function code
   51    01    91                               0C81  2984          CMPB    #XG_FC_V_STRT_LIN,R1                ; If EQL then start the line
         20    13                               0C84  2985          BEQL    10$
         0A    10                               0C86  2986          BSBB    START_DISP                          ; Branch to case
51    44 A5    D0                               0C88  2987          MOVL    UCB$L_DEVDEPEND(R5),R1              ; Set device characteristics
                                                0C8C  2988          REQCOM                                      ; Complete the request
                                                0C92  2989
                                                0C92  2990 START_DISP:
                                                0C92  2991          $DISPATCH       R1,TYPE=B,-
                                                0C92  2992          <-      ; Function              Action
                                                0C92  2993                  <XG_FC_V_STRT_CIR       START_CIRCUIT>,-
                                                0C92  2994                  <XG_FC_V_STRT_LIN               5$>,-
                                                0C92  2995                  <XG_FC_V_STOP_CIR       SHUTDOWN_CIRCUIT>,-
                                                0C92  2996                  <XG_FC_V_STOP_LIN       SHUTDOWN_LINE>,-
                                                0C92  2997                  <XG_FC_V_READ_MODEM     READ_MODEM>,-
                                                0C92  2998          >
                                                0CA2  2999
                                                0CA2  3000 5$:      BUG_CHECK NOBUFPCKT,FATAL                   ; Anything else is fatal
                                                0CA6  3001
      03AC    31                                0CA6  3002 10$:     BRW     START_LINE                          ; Branch to start the line
```

```
                0CA9  3003
                0CA9  3004          .DSABL  LSB
                0CA9  3005
```

```
                        0CA9  3007          .SBTTL  START_TRANSMIT - Start transmit I/O
                        0CA9  3008 ;++
                        0CA9  3009 ; START_TRANSMIT - Start transmit I/O
                        0CA9  3010 ;
                        0CA9  3011 ; FUNCTIONAL DESCRIPTION:
                        0CA9  3012 ;
                        0CA9  3013 ; This routine is entered to start a transmit I/O operation. It will call the
                        0CA9  3014 ; routine LOAD_XMT_MPR to see if there is a transmit to give the board. If a
                        0CA9  3015 ; transmit is loaded, this routine sets the enable bit on the board to initiate
                        0CA9  3016 ; sending of the data.
                        0CA9  3017 ;
                        0CA9  3018 ;
                        0CA9  3019 ; **** NOTE ****
                        0CA9  3020 ;
                        0CA9  3021 ; Must be sure when setting up the indirect register in misc register
                        0CA9  3022 ; that Master Reset is not accidentially set!!!!
                        0CA9  3023 ;
                        0CA9  3024 ;
                        0CA9  3025 ; INPUTS:
                        0CA9  3026 ;       R1 = Status information from routine calling START_TRANSMIT
                        0CA9  3027 ;            More specifically it is an interface between the protocol
                        0CA9  3028 ;            timer routine and this routine to tell start transmit that
                        0CA9  3029 ;            the protocol timer has expired therefore send the message.
                        0CA9  3030 ;       R5 = UCB ADDRESS
                        0CA9  3031 ;
                        0CA9  3032 ; OUTPUTS:
                        0CA9  3033 ;
                        0CA9  3034 ;       NONE
                        0CA9  3035 ;
                        0CA9  3036 ;       *** NOTE ***
                        0CA9  3037 ;
                        0CA9  3038 ; If the device is running in LAPB 30$ should never be called!
                        0CA9  3039 ;--
                        0CA9  3040 START_TRANSMIT:
                        0CA9  3041 ;
                        0CA9  3042 ; If the DDCMP timer has expired a message may need to be sent
                        0CA9  3043 ;
   05 51    0F    E0    0CA9  3044          BBS     #DLK$V_TMREXPD,R1,5$
            00    E0    0CAD  3045          BBS     #XG_DS_V_XMTING,-                       ; If BS then the xmt'r is off
      41 68 A5          0CAF  3046                  UCB$W_DEVSTS(R5),10$
 0120 C5    06    91    0CB2  3047 5$:      CMPB    #XG$C_DRPCTS,UCB$B_XG_XSTATE(R5) ; If EQL CTS has not gone away
            3A    13    0CB7  3048          BEQL    10$                                     ; do not bring up RTS until
                        0CB9  3049                                                          ; we are sure CTS has been
                        0CB9  3050                                                          ; dropped - for real modems
                        0CB9  3051          DSBINT  UCB$B_DIPL(R5)                          ; Disable device interrupts
                        0CC0  3052
                        0CC0  3053          ASSUME XG$C_PRIXMT EQ 0
                        0CC0  3054
2B 68 A5    0A    E0    0CC0  3055          BBS     #XG_DS_V_XMT_DONEP,UCB$W_DEVSTS(R5),9$ ; If BS do not q new xmt
            53    D4    0CC5  3056          CLRL    R3                                      ; Set to set up prim BACC
         52 0A    9A    0CC7  3057          MOVZBL  #XG$C_PRI_XMT,R2                        ; Set ind reg addr of prim
          006B    30    0CCA  3058          BSBW    LOAD_XMT_MPR                            ; Load map registers and BACC
        30 50    E9    0CCD  3059          BLBC    R0,30$                                  ; No tranmsit given to board
                        0CD0  3060          SETBIT  #XG_DS_V_XMT_DONEP,UCB$W_DEVSTS(R5) ; Set indicating that the
                        0CD5  3061                                                          ; XMTCSR are is use
                        0CD5  3062          SETIPL                                          ; Sync to highest IPL
1A 64 A5    05    E0    0CD8  3063          BBS     #UCB$V_POWER,UCB$W_STS(R5),20$  ; If BS then power fail occured
```

```
                02 A4    01    A8  0CDD  3064 8$:      BISW    #XG$M_ENABLE,XG$C_XMT_CSR(R4)   ; Else enable transmitter
          50    00000000'GF    D0  0CE1  3065          MOVL    G^EXE$GL_ABSTIM,R0              ; get current time
  0104 C5    0108 C5    50    C1  0CE8  3066          ADDL3   R0,UCB$L_XG_XMT_TIMEOUT(R5),UCB$L_XG_XMTEND(R5) ; Set timeout time
                                0CF0  3067 9$:        ENBINT                                  ; Enable interrupts
                     50    01  3C  0CF3  3068 10$:     MOVZWL  S^#SS$_NORMAL,R0                ; Set status
                                05  0CF6  3069          RSB
                                    0CF7  3070
                                    0CF7  3071 20$:     ENBINT                                  ; Enable interrupts
                50   0364 8F    3C  0CFA  3072          MOVZWL  #SS$_POWERFAIL,R0               ; Set status
                                05  0CFF  3073          RSB
                                    0D00  3074
                                    0D00  3075 30$:     ENBINT                                  ; Enable interrupts
                51   0167 C5    D0  0D03  3076          MOVL    UCB$B_XG_PROTYPE(R5),R1         ; Get protocol type
                                    0D08  3077          CASE    R1,TYPE=B,<-
                                    0D08  3078                  35$,-                           ; LAPB
                                    0D08  3079                  35$,-                           ; Bisync
                                    0D08  3080                  >                               ; Fall thru on DDCMP
                     50    95  0D10  3081          TSTB    R0                              ; If EQL XMTer is idle
                     20    12  0D12  3082          BNEQ    35$                             ; Else wait for the interrupt
                                0D14  3083                                                  ; from CTS being asserted
                     56    02  9A  0D14  3084          MOVZBL  #DLK$C_XMTMSG,R6                ; Inform DDCMP that the last
                     57    02  9A  0D17  3085          MOVZBL  #DLK$M_QEMPTY,R7                ; msg on queue was sent
                     55    DD  0D1A  3086          PUSHL   R5                              ; Save register
                55   00D0 C5    D0  0D1C  3087          MOVL    UCB$A_XG_PRO_BUFFER(R5),R5      ; Get addr of start of TFB
                        F2DC'   30  0D21  3088          BSBW    DDCMP                           ; Branch to protocol
                           55 8ED0  0D24  3089          POPL    R5                              ; Restore register
                                    0D27  3090
                                    0D27  3091          ASSUME DLK$V_MSGSENT EQ 0
                     56    02    91  0D27  3092          CMPB    #DLK$C_XMTMSG,R6                ; If NEQ no ACK to be xmted
                           08    12  0D2A  3093          BNEQ    35$
                  05 57    E9  0D2C  3094          BLBC    R7,35$                          ; If BC no  ACK need be sent
                     51    D4  0D2F  3095          CLRL    R1                              ; Set no status info
                   FF75    31  0D31  3096          BRW     START_TRANSMIT                  ; Else send the ACK
50                 01    3C  0D34  3097 35$:     MOVZWL  S^#SS$_NORMAL,R0                ; Set status
                           05  0D37  3098          RSB
                                0D38  3099
```

```
                        0D38  3101
                        0D38  3102          .SBTTL  Load Mapping registers and give to COMBO
                        0D38  3103 ;++
                        0D38  3104 ;LOAD_XMT_MPR - Load transmit mapping registers
                        0D38  3105 ;
                        0D38  3106 ; This routine gets a transmit out of the queue and sets up the mapping
                        0D38  3107 ; registers and BACC to give to COMBO. For simplicity only the primary
                        0D38  3108 ; buffer and character count is used. The tansmitter is allowed to be in
                        0D38  3109 ; three different states. The stopped state when there is no data to be
                        0D38  3110 ; transmitted. The wait for clear to send state when the transmitter
                        0D38  3111 ; has aserted request to send and is waiting for clear to send to come back.
                        0D38  3112 ; Finally the xmitting state when the transmitter has control of the line
                        0D38  3113 ; and can send data.
                        0D38  3114 ;
                        0D38  3115 ; This routine also does modem signal processing. For half duplex and tributary
                        0D38  3116 ; lines it sets RTS then puts the driver in a "Wait" for CTS to come back. If
                        0D38  3117 ; CTS dosen't come up in 1 microsecond the driver continues processing and
                        0D38  3118 ; expects a special CTS timer or the device timer routine to watch for CTS to
                        0D38  3119 ; come back. If the timer routine doesn't see CTS in a reasonable amount of
                        0D38  3120 ; time (2 to 3 seconds) and the line is half duplex point to point , a fatal
                        0D38  3121 ; error is assumed. If the line is in trib then the timer will wait indefinitly
                        0D38  3122 ; for CTS to come high.If the line is in loopback these signals are not
                        0D38  3123 ; checked. If the line is in full duplex RTS is never dropped.
                        0D38  3124 ;
                        0D38  3125 ;
                        0D38  3126 ;         INPUTS  R2 = Primary/secondary register to use
                        0D38  3127 ;                 R3 = Vector slot to use
                        0D38  3128 ;                 R5 = Address of the UCB
                        0D38  3129 ;                 IPL = DIPL
                        0D38  3130 ;
                        0D38  3131 ;         OUTPUT
                        0D38  3132 ;                 R4 = CSR address
                        0D38  3133 ;--
                        0D38  3134          .ENABL LSB
                        0D38  3135 LOAD_XMT_MPR:
    0140 8F    BB       0D38  3136          PUSHR   #^M<R6,R8>
  51   24 A5   D0       0D3C  3137          MOVL    UCB$L_CRB(R5),R1                ; Get the address of the CRB
  54   2C B1   D0       0D40  3138          MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R4    ; Get CSR address
56   00D0 C5   D0       0D44  3139          MOVL    UCB$A_XG_PRO_BUFFER(R5),R6      ; Set protocol buffer address
  58   20 B6   0F       0D49  3140          REMQUE  @TF$Q_CTLQ(R6),R8               ; Get next control message
         03   1D       0D4D  3141          BVS     1$
       0079   31       0D4F  3142          BRW     10$
  58   30 B6   0F       0D52  3143 1$:      REMQUE  @TF$Q_XMTQ(R6),R8               ; Get the next message to XMT
         18   1C       0D56  3144          BVC     2$                              ; If VC then data to give COMBO
    0120 C5   94       0D58  3145          CLRB    UCB$B_XG_XSTATE(R5)             ; Set XMTer IDLE
         50   D4       0D5C  3146          CLRL    R0                              ; Clear normal status
    0140 8F   BA       0D5E  3147          POPR    #^M<R6,R8>                      ; Restore the reg
              05       0D62  3148          RSB
                       0D63  3149 ;
                       0D63  3150 ; Return from CTS start timer
                       0D63  3151 ;
                       0D63  3152 CTSRET:
  0B A5   01   90      0D63  3153          MOVB    #TQE$C_SSSNGL,TQE$B_RQTYPE(R5)  ; Set single shot timer
         55 8ED0       0D67  3154          POPL    R5                              ; Restore R5
50   0120 C5   90      0D6A  3155          MOVB    UCB$B_XG_XSTATE(R5),R0          ; Set R0 for return
              05       0D6F  3156          RSB
                       0D70  3157
```

```
                13   0A A8   91   0D70  3158 2$:     CMPB    XMTQ$B_BUFTYP(R8),S^#DYN$C_BUFIO ; If EQL then buff ok
                         55   13   0D74  3159         BEQL    10$
                                   0D76  3160         BUG_CHECK       NOBUFPCKT,FATAL
                                   0D7A  3161
              0120 C5    04   91   0D7A  3162 4$:     CMPB    #XG$C_SWFCTS,UCB$B_XG_XSTATE(R5) ; If EQL then special wait
                         07   13   0D7F  3163         BEQL    5$                               ; for CTS state from CTS TQE
                                   0D81  3164
                                   0D81  3165         .IF     DF CTS$$$
                                   0D81  3166         pushl   r0
                                   0D81  3167         addl3   #1,ucb$b_cts_last(r5),r0         ; get next free slot
                                   0D81  3168         bicl    #^c<cts_size-1>,r0               ; make mod of cts_size
                                   0D81  3169         movb    r0,ucb$b_cts_last(r5)            ; set new location to incr
                                   0D81  3170         clrb    ucb$b_cts_buf(r5)[r0]            ; make sure loc is zero
                                   0D81  3171         popl    r0
                                   0D81  3172         .ENDC                                    ; DF CTS$$$ end
                                   0D81  3173
                    0155 C5    90   0D81  3174         MOVB    UCB$B_XG_TIMEOUT(R5),-           ; Set number of seconds to wait
                    0121 C5         0D85  3175                 UCB$B_XG_WFCTS_SEC(R5)           ; for CTS to be detected
              06 1F A8    02   E1   0D88  3176 5$:     BBC     #XMTQ$V_CONTROL,XMTQ$B_FLAG(R8),6$ ; If cntrl msg put on CTLQ
                 20 A6    68   0E   0D8D  3177         INSQUE  (R8),TF$Q_CTLQ(R6)               ; Put the CTL msg back on queue
                          0E   11   0D91  3178         BRB     8$                               ; Branch to start up tiemr
                13   0A A8   91   0D93  3179 6$:     CMPB    XMTQ$B_BUFTYP(R8),S^#DYN$C_BUFIO ; If EQL then buff ok
                         04   13   0D97  3180         BEQL    7$
                                   0D99  3181         BUG_CHECK       NOBUFPCKT,FATAL
                 30 A6    68   0E   0D9D  3182 7$:     INSQUE  (R8),TF$Q_XMTQ(R6)               ; Put the XMT back on queue
                    50    02   9A   0DA1  3183 8$:     MOVZBL  #XG$C_WFCTS,R0                   ; Set status in R0
              0120 C5     50   90   0DA4  3184         MOVB    R0,UCB$B_XG_XSTATE(R5)           ; Set new XMTer state
                    0140  8F   BA   0DA9  3185         POPR    #^M<R6,R8>                       ; Restore regs
           18 68 A5      0E   E2   0DAD  3186         BBSS    #XG_DS_V_CTSTQE_RUN,UCB$W_DEVSTS(R5),9$ ; If BS timer is q'd
                         55   DD   0DB2  3187         PUSHL   R5                               ; Save R5 across fork
                55   0094 C5   DE   0DB4  3188         MOVAL   UCB$L_XG_TQE(R5),R5              ; Get the CTS TQE block
                  0B A5      06   90   0DB9  3189         MOVB    #IPL$_QUEUEAST,TQE$B_RQTYPE(R5) ; Set IPL of fork process
                        A3   AF   9F   0DBD  3190         PUSHAB  B^CTSRET                         ; Set return address
                     0EC9'CF      3F   0DC0  3191         PUSHAW  W^START_TIMER                    ; Set addr of CTS TQE start
                00000000'GF      17   0DC4  3192         JMP     G^EXE$FORK
                                   0DCA  3193
                                05   0DCA  3194 9$:     RSB
                                   0DCB  3195
                                   0DCB  3196 ; Set RTS and wait for CTS
                                   0DCB  3197
                         01   E0   0DCB  3198 10$:    BBS     #XM$V_CHR_LOOPB,-                ; If BS then loopb don't check
                    53 44 A5      0DCD  3199                 UCB$L_DEVDEPEND(R5),12$          ; modem signals
              0150 C5     05   91   0DD0  3200         CMPB    #NMA$C_LINPR_LAPB,UCB$B_XG_PRO(R5) ; If EQL LAPB don't check
                          4C   13   0DD5  3201         BEQL    12$                              ; modem signals
              0120 C5     04   91   0DD7  3202         CMPB    #XG$C_SWFCTS,UCB$B_XG_XSTATE(R5) ; If EQL then special wait
                          17   13   0DDC  3203         BEQL    13$                              ; for CTS state from CTS TQE
                    0120 C5    95   0DDE  3204         TSTB    UCB$B_XG_XSTATE(R5)              ; If NEQ then XMTer not idle
                          44   12   0DE2  3205         BNEQ    15$
                 04 A4    04   B0   0DE4  3206 11$:    MOVW    #XG$C_MODEM,XG$C_MISC_REG(R4)    ; Get the indr modem reg
           06 A4      1000 8F   A8   0DE8  3207         BISW    #XG$M_RTS,XG$C_IND_ADDR(R4)      ; Set RTS
              02 44 A5    02   E1   0DEE  3208         BBC     #XM$V_CHR_HDPLX,UCB$L_DEVDEPEND(R5),13$
                                   0DF3  3209 ;       BITW    #XG$M_CARRIER,XG$C_IND_ADDR(R4)
                                   0DF3  32'0 ;       BNEQ    14$
                          85   11   0DF3  3 1         BRB     4$                               ; Always wait for CTS
                                   0DF5  3212
                 04 A4    04   B0   0DF5  3213 13$:    MOVW    #XG$C_MODEM,XG$C_MISC_REG(R4)    ; Get the indr modem reg
                                   0DF9  3214         TIMEWAIT #1,#XG$M_CTS,XG$C_IND_ADDR(R4),W
```

```
                03 50    E8  0E1D  3215          BLBS    R0,12$                          ; If LBS then CTS was detected
                 FF57    31  0E20  3216 14$:     BRW     4$                              ; Else, start timers
                         0E23  3217
                   01    90  0E23  3218 12$:     MOVB    #XG$C_XMTING,-                  ; Set XMTer state on
              0120 C5        0E25  3219                  UCB$B_XG_XSTATE(R5)
         0120 C5   02    91  0E28  3220 15$:     CMPB    #XG$C_WFCTS,UCB$B_XG_XSTATE(R5) ; If EQL then XMTer waiting for
                   03    12  0E2D  3221          BNEQ    17$                             ; CTS
                 FF56    31  0E2F  3222          BRW     5$                              ; Else, check timers
                         0E32  3223
                         0E32  3224 ; Else mark the slot inuse and create buffer address / character count image
                         0E32  3225
                   14    BB  0E32  3226 17$:     PUSHR   #^M<R2,R4>                      ; Save registers
            1F A8   01    8A  0E34  3227          BICB2   #XMTQ$M_ONQUEUE,XMTQ$B_FLAG(R8) ; Mark that the msg was sent
         03 1F A8   02    E0  0E38  3228          BBS     #XMTQ$V_CONTROL,XMTQ$B_FLAG(R8),18$ ; If control msg don't incr
                         0E3D  3229 ;
                         0E3D  3230 ; The INCB applies only when the driver is running in DDCMP mode. The
                         0E3D  3231 ;  next instruction is a NOOP when running in BISYNC and LAPB mode. However,
                         0E3D  3232 ; since DDCMP is the protocol that needs to run the fastest, we opted
                         0E3D  3233 ; note to code around this instruction, but instead just allocate a byte
                         0E3D  3234 ; in each of the protocol buffers to allow falling thru this instruction.
                         0E3D  3235 ; When reading the code you will notice that there is an XQCNT location
                         0E3D  3236 ; in the data structures of each of the buffers allocated to the protocols.
                         0E3D  3237 ;
                10 A6    96  0E3D  3238          INCB    TF$B_XQCNT(R6)                  ; Incr implies 1 less msg on Q
                   01    E5  0E40  3239 18$:     BBCC    #XMTQ$V_SELECT,-                ; If BC then select flag not
                04 1F A8       0E42  3240                  XMTQ$B_FLAG(R8),20$             ; set don't turn link
                   01    A8  0E45  3241          BISW2   #XG_DS_M_XMTING,-               ; Set transmitter off but
                68 A5          0E47  3242                  UCB$W_DEVSTS(R5)                ; finish sending this msg
            52    24 A8    DE  0E49  3243 20$:     MOVAL   XMTQ$B_MSGHDR(R8),R2            ; Get address of data to send
         54   0110 C543    DE  0E4D  3244          MOVAL   UCB$Z_XG_VECTOR(R5)[R3],R4      ; Get the slot to use
            14 A8    52    D0  0E53  3245          MOVL    R2,XMTQ$L_BACC(R8)              ; Get the message offset
                1A A8    B0  0E57  3246          MOVW    XMTQ$W_MSGSIZE(R8),-            ; Set the char count
                16 A8          0E5A  3247                  XMTQ$L_BACC+2(R8)
   14 A8   07   09    64    F0  0E5C  3248          INSV    (R4),#9,#7,XMTQ$L_BACC(R8)      ; Insert the map register
      50   64   02    07    EF  0E62  3249          EXTZV   #7,#2,(R4),R0                   ; Get the high two bits
   14 A8   02   1E    50    F0  0E67  3250          INSV    R0,#30,#2,XMTQ$L_BACC(R8)       ; Insert into high bits
              00FC C543    D5  0E6D  3251          TSTL    UCB$Z_XG_XMT_INPR(R5)[R3]       ; Check to make sure we aren't
                   51    12  0E72  3252          BNEQ    40$                             ; overwriting a XMT inprogress
          00FC C543    58    D0  0E74  3253          MOVL    R8,UCB$Z_XG_XMT_INPR(R5)[R3]    ; Store the addr of buff'r sent
                0164 C5    96  0E7A  3254          INCB    UCB$B_XG_XMTCNT(R5)             ; Incr the no of outstndng XMTs
                         0E7E  3255
                         0E7E  3256 ; Build UBA map registers and load the map
                         0E7E  3257
   7C A5   52    FE00 8F    AB  0E7E  3258          BICW3   #^C<VA$M_BYTE>,R2,-             ; Store the message offset
                         0E85  3259                  UCB$W_BOFF(R5)
      52   52    15   09    EF  0E85  3260          EXTZV   #VA$V_VPN,#VA$S_VPN,R2,R2       ; Get the virtual page number
       50   00000000'GF    D0  0E8A  3261          MOVL    G^MMG$GL_SPTBASE,R0             ; Get the address of SPT entry
            78 A5   6042    DE  0E91  3262          MOVAL   (R0)[R2],UCB$L_SVAPTE(R5)       ; Set address of SPT entry
                   1A A8    B0  0E96  3263          MOVW    XMTQ$W_MSGSIZE(R8),-            ; Get the byte count
                   7E A5          0E99  3264                  UCB$W_BCNT(R5)
              51   24 A5    D0  0E9B  3265          MOVL    UCB$L_CRB(R5),R1                ; Get the address of the CRB
                   64       D0  0E9F  3266          MOVL    (R4),-                          ; Set starting map register
                   34 A1          0EA1  3267                  CRB$L_INTD+VEC$W_MAPREG(R1)
               0C000000'GF    16  0EA3  3268          JSB     G^IOC$LOADUBAMAPA               ; Load the map register
                      14    BA  0EA9  3269          POPR    #^M<R2,R4>                      ; Restore the registers
                04 A4   52    B0  0EAB  3270          MOVW    R2,XG$C_MISC_REG(R4)            ; Load into indirect register
                   14 A8    B0  0EAF  3271          MOVW    XMTQ$L_BACC(R8),-               ; Load buffer address
```

```
                06 A4           0EB2  3272                    XG$C_IND_ADDR(R4)
                16 A8    B0     0EB4  3273            MOVW    XMTQ$L_BACC+2(R8),-             ; and character count
                06 A4           0EB7  3274                    XG$C_IND_ADDR(R4)
          02 A4    04    AA     0EB9  3275            BICW2   #XG$M_PRM_SEC,XG$C_XMT_CSR(R4)  ; Since only one is ever loaded
                                0EBD  3276                                                    ; be sure this bit is always
                                0EBD  3277                                                    ; cleared
             50    01    3C     0EBD  3278            MOVZWL  S^#SS$_NORMAL,R0                ; Set BACC loaded
             0140 8F     BA     0EC0  3279 30$:       POPR    #^M<R6,R8>
                         05     0EC4  3280            RSB
                                0EC5  3281            .DSABL  LSB
                                0EC5  3282
                                0EC5  3283 40$:       BUG_CHECK       NOBUFPCKT,FATAL
                                0EC9  3284
```

```
                                0EC9  3286
                                0EC9  3287 ;++
                                0EC9  3288 ;START_TIMER - Start up for CTS TQE
                                0EC9  3289 ;
                                0EC9  3290 ; This routine starts up the CTS TQE which times the return of CTS after
                                0EC9  3291 ; setting RTS.
                                0EC9  3292 ;
                                0EC9  3293 ;       INPUTS  R5 = Adress of TQE block
                                0EC9  3294 ;               IPL = Queueast IPL
                                0EC9  3295 ;
                                0EC9  3296 ;       OUTPUT: Entry added to timer queue
                                0EC9  3297 ;
                                0EC9  3298 ;--
                                0EC9  3299 START_TIMER:
                         3F     BB  0EC9  3300         PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; Save the registers
      0C A5  00001D30'GF        DE  0ECB  3301         MOVAL   G^CTS_TIMER,TQE$L_FPC(R5)       ; Set the addr of timer wakeup
         50  00000000'GF        7D  0ED3  3302         MOVQ    G^EXE$GQ_SYSTIME,R0             ; Set time
                                    0EDA  3303         DSBINT  #IPL$_TIMER                     ; Sync to timer IPL
             00000000'GF        16  0EE0  3304         JSB     G^EXE$INSTIMQ                   ; Give to timer queue
                                    0EE6  3305         ENBINT
                         3F     BA  0EE9  3306         POPR    #^M<R0,R1,R2,R3,R4,R5>
                                05  0EEB  3307         RSB
```

```
                                0EEC  3309          .SBTTL  READ_MODEM - Read device modem register
                                0EEC  3310 ;++
                                0EEC  3311 ;READ_MODEM - Read device modem register
                                0EEC  3312 ;
                                0EEC  3313 ; This routine returns the device modem register in the buffer passed.
                                0EEC  3314 ; These are set according to the XMDEF's for modem bits.
                                0EEC  3315 ;
                                0EEC  3316 ;       Inputs:
                                0EEC  3317 ;               R3 = IRP address
                                0EEC  3318 ;               R5 = UCB address
                                0EEC  3319 ;
                                0EEC  3320 ;--
                                0EEC  3321 READ_MODEM:
 06 68 A5    02    E0           0EEC  3322          BBS     #XG_DS_V_INITED,UCB$W_DEVSTS(R5),5$ ; If BS then read register
  50   20D4 8F    3C           0EF1  3323          MOVZWL  #SS$_DEVINACT,R0                    ; Else return device inactive
                  05           0EF6  3324          RSB
                               0EF7  3325
                     54    DD  0EF7  3326 5$:      PUSHL   R4
            54    24 A5    D0  0EF9  3327          MOVL    UCB$L_CRB(R5),R4                    ; Get CRB address
            54    2C B4    D0  0EFD  3328          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4        ; Get device CSR
                               0F01  3329          DSBINT  UCB$B_DIPL(R5)                      ; Sync to device IPL
            04 A4    04    B0  0F08  3330          MOVW    #XG$C_MODEM,XG$C_MISC_REG(R4)       ; Set to read modem register
            51    06 A4    3C  0F0C  3331          MOVZWL  XG$C_IND_ADDR(R4),R1                ; Read modem register
                               0F10  3332          ENBINT                                      ; Enable interrupts
            54    2C A3    D0  0F13  3333          MOVL    IRP$L_SVAPTE(R3),R4                 ; Get address of buffer
                     50    D4  0F17  3334          CLRL    R0
            04 51    04    E1  0F19  3335          BBC     #XG$V_CTS,R1,10$                    ; BC CTS is not set
                               0F1D  3336          SETBIT  #XM$V_MDM_CTS,R0                    ; Set CTS for return
            04 51    05    E1  0F21  3337 10$:     BBC     #XG$V_CARRIER,R1,20$
                               0F25  3338          SETBIT  #XM$V_MDM_CARRDET,R0
            04 51    06    E1  0F29  3339 20$:     BBC     #XG$V_RING_IND,R1,30$
                               0F2D  3340          SETBIT  #XM$V_MDM_RING,R0
            04 51    07    E1  0F31  3341 30$:     BBC     #XG$V_DSR,R1,40$
                               0F35  3342          SETBIT  #XM$V_MDM_DSR,R0
            04 51    09    E1  0F39  3343 40$:     BBC     #XG$V_DTR,R1,50$
                               0F3D  3344          SETBIT  #XM$V_MDM_DTR,R0
            04 51    0C    E1  0F41  3345 50$:     BBC     #XG$V_RTS,R1,60$
                               0F45  3346          SETBIT  #XM$V_MDM_RTS,R0
            0C A4    50    D0  0F49  3347 60$:     MOVL    R0,P2B_T_DATA(R4)                   ; Return modem signals
                  54 8ED0    POPL  0F4D  3348          POPL    R4
                50    01    3C  0F50  3349          MOVZWL  S^#SS$_NORMAL,R0
                            05  0F53  3350          RSB
                               0F54  3351
```

```
                     0F54  3353          .SBTTL  CLEAN - Abort all outstanding XMT's (All IO's for BISYNC)
                     0F54  3354 ;++
                     0F54  3355 ; CLEAN - Abort all outstanding XMT's (all IO's for BISYNC)
                     0F54  3356 ;
                     0F54  3357 ; This routine aborts all outstanding XMT's on the device when running
                     0F54  3358 ; in LAPB mode and all outstanding IO's when running in BISYNC protocol
                     0F54  3359 ; mode. To do this it syncs to device IPL takes all XMT's off
                     0F54  3360 ; the queues, puts them temporarily on another queue, then check for
                     0F54  3361 ; in XMT inprogress. If an XMT is in progress, we can not just stop it
                     0F54  3362 ; the board won't like it, therefore we set a bit saying when
                     0F54  3363 ; this XMT ends complete it with an abort status and complete all the
                     0F54  3364 ; outstanding XMT's then complete the CLEAN request. If no XMT is
                     0F54  3365 ; in progress then all XMT's are taken from the temporary queue and
                     0F54  3366 ; complete via IOPOST. Remember they are completed at FIPL, therefore
                     0F54  3367 ; if we took them from the device queue at a lowered IPL, completed the xmt
                     0F54  3368 ; then raised IPL; we may allow xmt's to complete normally.
                     0F54  3369 ;
                     0F54  3370 ; Note any XMT in the post queue is complete in error. This is done because
                     0F54  3371 ; there is no way to determine if the clean was issued before or after
                     0F54  3372 ; the XMT actually completes. This method also provides less risk of getting
                     0F54  3373 ; XMT's completed out of order
                     0F54  3374 ;
                     0F54  3375 ;       INPUTS: R3 = IRP address
                     0F54  3376 ;               R5 = UCB address
                     0F54  3377 ;               IPL = FIPL
                     0F54  3378 ;
                     0F54  3379 ;       OUTPUTS R0 = Status
                     0F54  3380 ;               R3 = IRP address
                     0F54  3381 ;               R5 = UCB address
                     0F54  3382 ;
                     0F54  3383 ; Clean_fork_entry
                     0F54  3384 ;       Input   R1 = Allocated buffer address
                     0F54  3385 ;               R3 = IRP of outstanding IO
                     0F54  3386 ;               R4 = Address of the protoocl work buffer
                     0F54  3387 ;               R5 = UCB address
                     0F54  3388 ;               IPL = FIPL
                     0F54  3389 ;
                     0F54  3390 ;--
                     0F54  3391          .ENABL LSB
                     0F54  3392 CLEAN:
0168 C5    53   D0  0F54  3393          MOVL    R3,UCB$A_XG_CLEAN(R5)           ; Save CLEAN IRP address
          53   D4  0F59  3394          CLRL    R3                              ; Clear R3
                     0F5B  3395          DSBINT  UCB$B_DIPL(R5)                  ; Sync to clear device queue
54   00D0 C5   D0  0F62  3396          MOVL    UCB$A_XG_PRO_BUFFER(R5),R4      ; Set protocol buffer address
                     0F67  3397          ASSUME  XG$C_PROTYPE_LAPB EQ 0
     0167 C5   95  0F67  3398          TSTB    UCB$B_XG_PROTYPE(R5)            ; If neql then must be bisync
          0C   12  0F6B  3399          BNEQ    15$                             ; so take xmts off bisycn queue
  51   20 B4   0F  0F6D  3400 10$:     REMQUE  @LAPB$Q_XMTQ(R4),R1             ; Get next xmt
          12   1D  0F71  3401          BVS     20$
  2C B4   61   0E  0F73  3402          INSQUE  (R1),@LAPB$Q_CLEANQ+4(R4)       ; Put on clean queue
          F4   11  0F77  3403          BRB     10$                             ; get next xmt
                     0F79  3404
  51   30 B4   0F  0F79  3405 15$:     REMQUE  @BISYNC$Q_XMTQ(R4),R1           ; Get next xmt
          06   1D  0F7D  3406          BVS     20$
  2C B4   61   0E  0F7F  3407          INSQUE  (R1),@BISYNC$Q_CLEANQ+4(R4)     ; Put on clean queue
          F4   11  0F83  3408          BRB     15$                             ; get next xmt
                     0F85  3409
```

```
                51   24 A5   D0  0F85  3410 20$:    MOVL    UCB$L_CRB(R5),R1                        ; Get CRB
                51   2C B1   D0  0F89  3411         MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R1            ; Get CSR address
                02 A1   01   AA  0F8D  3412         BICW2   #XG$M_ENABLE,XG$C_XMT_CSR(R1)           ; Clear the enable bit
                   0164 C5   95  0F91  3413         TSTB    UCB$B_XG_XMTCNT(R5)                     ; IF NEQ zero xmt inpr
                        76   12  0F95  3414         BNEQ    50$                                     ; Branch to await completion
                             00  0F97  3415         ENBINT                                          ; Set to FIPL
                             00  0F9A  3416
                             00  0F9A  3417 CLEAN_FORK_ENTRY:
                0167 C5   02   91  0F9A  3418         CMPB    #XG$C_PROTYPE_DDCMP,UCB$B_XG_PROTYPE(R5) ; If EQL then BUG
                        7C   13  0F9F  3419         BEQL    60$
                        53   D5  0FA1  3420         TSTL    R3                                      ; IF EQL then not from FORK
                        07   13  0FA3  3421         BEQL    30$
                  38 A3   50   D0  0FA5  3422         MOVL    R0,IRP$L_MEDIA(R3)                      ; Status of IO is set in R0
                      0CED   30  0FA9  3423         BSBW    TRANSMIT_IO_DONE                        ; Complete req in R3
                50    00F4 C5   DE  0FAC  3424 30$:    MOVAL   UCB$Q_XG_POST(R5),R0                    ; Get the address of queue head
                     51   60   D0  0FB1  3425         MOVL    (R0),R1                                 ; Get the first entry
                     51   50   D1  0FB4  3426 32$:    CMPL    R0,R1                                   ; If equal than end of queue
                        22   13  0FB7  3427         BEQL    37$
                  0A A1   13   91  0FB9  3428         CMPB    S^#DYN$C_BUFIO,IRP$B_TYPE(R1)           ; If NEQ then not a transmit
                        17   12  0FBD  3429         BNEQ    35$                                     ; get next entry
                     51   61   0F  0FBF  3430         REMQUE  (R1),R1                                 ; Else complete the request
                53    0C A1   D0  0FC2  3431         MOVL    XMTQ$L_IRP(R1),R3                       ; Get address of IRP
                  38 A3   2C   3C  0FC6  3432         MOVZWL  #SS$_ABORT,IRP$L_MEDIA(R3)              ; Set abort status
                     52   61   D0  0FCA  3433         MOVL    (R1),R2                                 ; Get next entry on queue
                        15   BB  0FCD  3434         PUSHR   #^M<R0,R2,R4>                           ; Save addr start of q and
                             00  0FCF  3435                                                         ; next entry
                      0CC7   30  0FCF  3436         BSBW    TRANSMIT_IO_DONE                        ; Complete request
                        13   BA  0FD2  3437         POPR    #^M<R0,R1,R4>                           ; Restore address in of q head
                             00  0FD4  3438                                                         ; and entry
                        DE   11  0FD4  3439         BRB     32$                                     ; branch to check if XMT
                     51   61   D0  0FD6  3440 35$:    MOVL    (R1),R1                                 ; Get next entry
                        D9   11  0FD9  3441         BRB     32$                                     ; branch to check for XMT
                  51    28 B4   0F  0FDB  3442 37$:    REMQUE  @LAPB$Q_CLEANQ(R4),R1                   ; Get buffer to complete
                        12   1D  0FDF  3443         BVS     40$                                     ; If VS then done
                53    0C A1   D0  0FE1  3444         MOVL    XMTQ$L_IRP(R1),R3                       ; Get IRP address
                  38 A3   2C   3C  0FE5  3445         MOVZWL  #SS$_ABORT,IRP$L_MEDIA(R3)              ; Set abort status
                        54   DD  0FE9  3446         PUSHL   R4
                      0CAB   30  0FEB  3447         BSBW    TRANSMIT_IO_DONE                        ; Complete the request
                        54 8ED0  0FEE  3448         POPL    R4
                        E8   11  0FF1  3449         BRB     37$                                     ; Get next buffer to complete
                0167 C5   01   91  0FF3  3450 40$:    CMPB    #XG$C_PROTYPE_BISYNC,UCB$B_XG_PROTYPE(R5) ; BISYNC?
                        03   12  0FF8  3451         BNEQ    45$                                     ; If NEQ then not
                      0024   30  0FFA  3452         BSBW    CLEAN_BISYNC                            ; Call to abort all IO'S
                53    0168 C5   D0  0FFD  3453 45$:    MOVL    UCB$A_XG_CLEAN(R5),R3                   ; Get REQ address
                   0168 C5   D4  1002  3454         CLRL    UCB$A_XG_CLEAN(R5)                      ; In case it gets called again
                  38 A3   01   3C  1006  3455         MOVZWL  S^#SS$_NORMAL,IRP$L_MEDIA(R3)           ; Set success
                      0C9E   31  100A  3456         BRW     IO_DONE
                             00  100D  3457
                             00  100D  3458 50$:    SETBIT  #XG_DS_V_CLEAN,UCB$W_DEVSTS(R5) ; Set to abort the xmt inpr
                             00  1012  3459         ENBINT                                          ; and other I/O and clean irp
                     50   01   3C  1015  3460         MOVZWL  S^#SS$_NORMAL,R0                        ; Set normal ret
                  51    44 A5   D0  1018  3461         MOVL    UCB$L_DEVDEPEND(R5),R1
                        05   00  101C  3462         RSB
                             00  101D  3463 60$:    BUG_CHECK       NOBUFPCKT,FATAL
                             00  1021  3464
                             00  1021  3465         .DSABL  LSB
```

```
                        1021  3467 ;++
                        1021  3468 ; CLEAN_BISYNC - Handle remainder of CLEAN work for BISYNC
                        1021  3469 ;
                        1021  3470 ; This routine aborts the remainder of the outstanding IO's from the driver,
                        1021  3471 ; to statify the BISYNC clean request. Unlike in LAPB the receive
                        1021  3472 ; IRP's must also be aborted when running in BISYNC mode. It also resets the
                        1021  3473 ; state (UCB$Q_XG_STATE_INFO) of the framming routine.
                        1021  3474 ;
                        1021  3475 ;       INPUTS  R4 = Address of protocol work buffer
                        1021  3476 ;               R5 = Address of UCB
                        1021  3477 ;
                        1021  3478 ;       OUTPUTS All outstanding receive IRPs are aborted
                        1021  3479 ;--
                        1021  3480 CLEAN_BISYNC:
                        1021  3481         CLRBIT  #XG_DS_V_XMTING,UCB$W_DEVSTS(R5) ; Set transmitter back on
            0FE3   30   1026  3482         BSBW    STOP_RCV_BISYNC                  ; Stop the device from rcving
              52   D5   1029  3483         TSTL    R2                               ; Branch if eql no buffer
              05   13   102B  3484         BEQL    10$                              ; to return to free list
00E8 D5       62   0E   102D  3485         INSQUE  (R2),@UCB$Q_XG_FREE+4(R5)        ; Return buffer to the list
53   00DC D5       0F   1032  3486 10$:     REMQUE  @UCB$Q_XG_RCVS(R5),R3            ; Abort all RCV IRP's
              0E   1D   1037  3487         BVS     20$
  38 A3       2C   3C   1039  3488         MOVZWL  #SS$_ABORT,IRP$L_MEDIA(R3)
              54   DD   103D  3489         PUSHL   R4
            0C69   30   103F  3490         BSBW    IO_DONE
              54 8ED0   1042  3491         POPL    R4
              EB   11   1045  3492         BRB     10$
                        1047  3493 20$:     DSBINT  UCB$B_DIPL(R5)
            0645   30   104E  3494         BSBW    START_RECEIVE_BISYNC             ; Startup receiver
                        1051  3495         ENBINT
                  05    1054  3496         RSB
                        1055  3497
```

```
                                1055  3499          .SBTTL  START - Start unit, device and/or protocol
                                1055  3500 ;++
                                1055  3501 ; START_LINE - Start unit
                                1055  3502 ;
                                1055  3503 ; Functional description:
                                1055  3504 ;
                                1055  3505 ; This routine initializes the UCB; allocates the map registers for
                                1055  3506 ; transmits and receives; and master clears the device.
                                1055  3507 ;
                                1055  3508 ; If a failure occurs the line shutdown sequence is entered.
                                1055  3509 ;
                                1055  3510 ; INPUTS:
                                1055  3511 ;
                                1055  3512 ;       R3 = I/O packet
                                1055  3513 ;       R5 = UCB address
                                1055  3514 ;
                                1055  3515 ; IMPLICIT INPUTS:
                                1055  3516 ;
                                1055  3517 ;       IRP$L_MEDIA contains a copy of the mode buffer specified by the user.
                                1055  3518 ;       IRP$W_QUOTA contains the quota taken from the user for the unit.
                                1055  3519 ;
                                1055  3520 ; OUTPUTS:
                                1055  3521 ;
                                1055  3522 ;       The request is completed.
                                1055  3523 ;--
                                1055  3524 START_LINE:                                ; Start protocol operation
                          02    E1    1055  3525          BBC     #XG_DS_V_INITED,-   ; If BC then device not inited
                       06 68    A5          1057  3526          UCB$W_DEVSTS(R5),5$     ; Branch to set up idle UCB
                  50    02C4    8F    3C    105A  3527          MOVZWL  #SS$_DEVACTIVE,R0   ; Set device active
                                05    105F  3528          RSB                         ; ... and return
                                      1060  3529
                      010E      C5    94    1060  3530 5$:     CLRB    UCB$B_XG_INUS(R5)           ; Clear rcv slots in use mask
                44 A5    FFF3F300    8F    CA    1064  3531          BICL    #XG$C_LINE_PAR,UCB$L_DEVDEPEND(R5) ; Reset all Read/Write flags
                   010C  C5    40    A3    B0    106C  3532          MOVW    IRP$W_QUOTA(R3),UCB$W_XG_QUOTA(R5) ; Set quota for RCV's
                         40    A3    B4    1072  3533          CLRW    IRP$W_QUOTA(R3)             ; Make quota consistent
                                      1075  3534
                                      1075  3535 ; Using the default buffer size and the minimum line speed set up the
                                      1075  3536 ; maximum amount of time to wait before timing out a transmit inprogress
                                      1075  3537 ; (that is after it has been given to the DMF32). This is to get around
                                      1075  3538 ; a problem where the DMF32 occasionaly loses clocking on the line and
                                      1075  3539 ; the transmit appears to get stuck. The alogorithm for this calculation
                                      1075  3540 ; is the following:
                                      1075  3541 ;
                                      1075  3542 ;       timeout_time = ((default_buffer_size * bits_per_char)/ minimum_baud_rate))+f
                                      1075  3543 ;
                                      1075  3544
                      50    42    A5    3C    1075  3545          MOVZWL  UCB$W_DEVBUFSIZ(R5),R0
                         51    08    9A    1079  3546          MOVZBL  #XG$C_BPC_DEFAULT,R1
                         51    50    C4    107C  3547          MULL2   R0,R1
                51    0000012C    8F    C6    107F  3543          DIVL2   #XG$C_BRG_DEFAULT,R1
                0108  C5    51    02    C1    1086  3549          ADDL3   #2,R1,UCB$L_XG_XMT_TIMEOUT(R5)
                                      108C  3550
                      0104      C5    D4    108C  3551          CLRL    UCB$L_XG_XMTEND(R5)         ; make sure timeout time is zero
                                      1090  3552
                                      1090  3553 ; Initialize the vector, which contains mapping information
                                      1090  3554 ; for each of the possible outstanding buffers.
                                      1090  3555
```

```
                50   04    9A  1090  3556          MOVZBL  #NUM_MAP_REG,R0                 ; Number of rcv vector slots
           51   0110 C5    DE  1093  3557          MOVAL   UCB$Z_XG_VECTOR(R5),R1          ; Get vector address
                     81    B4  1098  3558 10$:     CLRW    (R1)+                           ; Init data path
                81   01    AE  109A  3559          MNEGW   #1,(R1)+                        ; Indicate no map regs
                  F8 50    F5  109D  3560          SOBGTR  R0,10$                          ; Continue for entire vector
          012C C5   0C A3    D0  10A0  3561          MOVL    IRP$L_PID(R3),UCB$L_XG_PID(R5)  ; Save starter's PID
                    28 A3    B0  10A6  3562          MOVW    IRP$W_CHAN(R3),-                ; Save channel number
                  0130 C5        10A9  3563                  UCB$W_XG_CHANL(R5)
                       04    A8  10AC  3564          BISW    #XG_DS_M_INITED,-               ; Indicate UCB initialized
                    68 A5        10AE  3565                  UCB$W_DEVSTS(R5)
                               10B0  3566 ;
                               10B0  3567 ; Allocate map registers
                               10B0  3568 ;
                    42 A5    B0  10B0  3569          MOVW    UCB$W_DEVBUFSIZ(R5),-           ; Set up to alloc the map regs
                    7E A5        10B3  3570                  UCB$W_BCNT(R5)
          7C A5   01FF 8F    B0  10B5  3571          MOVW    #511,UCB$W_BOFF(R5)             ; Set worst case page cross
                54   24 A5    D0  10BB  3572          MOVL    UCB$L_CRB(R5),R4                ; Address unit CRB
                               10BF  3573
                               10BF  3574          ASSUME  VEC$B_NUMREG EQ VEC$W_MAPREG+2
                               10BF  3575          ASSUME  VEC$B_DATAPATH EQ VEC$B_NUMREG+1
                               10BF  3576
                    34 A4    D4  10BF  3577          CLRL    CRB$L_INTD+VEC$W_MAPREG(R4)     ; Clear map register + datapath
                  00C0 8F    BB  10C2  3578          PUSHR   #^M<R6,R7>                      ; Save regs
                56  0110 C5    DE  10C6  3579          MOVAL   UCB$Z_XG_VECTOR(R5),R6          ; Get vector address
                    57   04    9A  10CB  3580          MOVZBL  #NUM_MAP_REG,R7                 ; Get number of slots to use
                 00000000'GF   16  10CE  3581 15$:     JSB     G^IOC$ALOUBAMAP                 ; Get map regs
                       07 50    E9  10D4  3582          BLBC    R0,20$                          ; Br on error
                    34 A4    D0  10D7  3583          MOVL    CRB$L_INTD+VEC$W_MAPREG(R4),-   ; Save mapping info
                       86        10DA  3584                  (R6)+
                  F0 57    F5  10DB  3585          SOBGTR  R7,15$                          ; Continue until done
              00C0 8F    BA  10DE  3586 20$:     POPR    #^M<R6,R7>                      ; Restore regs
                0F 50    E9  10E2  3587          BLBC    R0,START_CTRL_ERROR             ; Branch LBC error on start up
                               10E5  3588 ;
                               10E5  3589 ; Master clear the device
                               10E5  3590 ;
          54    2C B4    D0  10E5  3591          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4    ; Get the CSR
          53    58 A5    D0  10E9  3592          MOVL    UCB$L_IRP(R5),R3                ; Pick up packet address
                               10ED  3593
                 00DB    30  10ED  3594          BSBW    AWAIT_UNIT                      ; Await the unit
             50    01    3C  10F0  3595          MOVZWL  S^#SS$_NORMAL,R0                ; Set normal return
                       05    10F3  3596          RSB
                               10F4  3597
                               10F4  3598 START_CTRL_ERROR:
                    50   DD  10F4  3599          PUSHL   R0                              ; Save abort reason
                  1005   30  10F6  3600          BSBW    SHUTDOWN_LINE                   ; Shut down the device
                    50 8ED0  10F9  3601          POPL    R0                              ; Restore abort reason
          51    44 A5    D0  10FC  3602          MOVL    UCB$L_DEVDEPEND(R5),R1          ; Set device charateristics
                               1100  3603          REQCOM                                  ; Complete the request
                               1106  3604
                               1106  3605
```

```
                                1106  3607
                                1106  3608 ;++
                                1106  3609 ; START_CIRCUIT- Start device and protocol
                                1106  3610 ;
                                1106  3611 ; Functional description:
                                1106  3612 ;
                                1106  3613 ; This routine sets up the protocol and device charateristics; sets up receive
                                1106  3614 ; buffers for the device; starts the protocol; and finally gives the first
                                1106  3615 ; transmit and receives to the board.
                                1106  3616 ;
                                1106  3617 ; If a failure occurs the circuit shutdown sequence is entered.
                                1106  3618 ;
                                1106  3619 ; INPUTS:
                                1106  3620 ;
                                1106  3621 ;       R3 = I/O packet
                                1106  3622 ;       R5 = UCB address
                                1106  3623 ;
                                1106  3624 ; IMPLICIT INPUTS:
                                1106  3625 ;
                                1106  3626 ;       IRP$L_MEDIA contains a copy of the mode buffer specified by the user.
                                1106  3627 ;
                                1106  3628 ; OUTPUTS:
                                1106  3629 ;
                                1106  3630 ;       The request is completed.
                                1106  3631 ;--
                                1106  3632 START_CIRCUIT:                                    ; Start the circuit
                          0B    E1  1106  3633          BBC     #XM$V_STS_ACTIVE,-          ; If BC then device and
                    06 44 A5        1108  3634                  UCB$L_DEVDEPEND(R5),5$      ; protocol are not active
                50   02C4 8F    3C  1108  3635          MOVZWL  #SS$_DEVACTIVE,R0           ; Return error
                             05     1110  3636          RSB
                                    1111  3637
                          02    E0  1111  3638 5$:      BBS     #XG_DS_V_INITED,-           ; If BS then line started
                    06 68 A5        1113  3639                  UCB$W_DEVSTS(R5),8$
                50   20D4 8F    3C  1116  3640          MOVZWL  #SS$_DEVINACT,R0            ; Else return device inactive
                             05     111B  3641          RSB
                                    111C  3642
                             AA     111C  3643 8$:      BICW    #^C<XG_DS_M_INITED!-        ; Reset bit fields
                                    111D  3644                  XG_DS_M_CTSTOE_RUN>,-
                 68 A5    BFFB 8F       111D  3645                  UCB$W_DEVSTS(R5)
             44 A5   FFFFF700 8F    CA  1122  3646          BICL    #XG$C_CIR_PAR,UCB$L_DEVDEPEND(R5) ; Reset all Read/Write flags
                                    112A  3647
                                    112A  3648          ASSUME  XG$C_IDLE EQ 0
                                    112A  3649          ASSUME  XG$C_XMTING EQ 1
                                    112A  3650          ASSUME  XG$C_WFCTS EQ 2
                                    112A  3651          ASSUME  XG$C_SWFCTS EQ 4
                                    112A  3652
                        0120 C5 94  112A  3653          CLRB    UCB$B_XG_XSTATE(R5)         ; Set XMTer state to idle
                        0164 C5 94  112E  3654          CLRB    UCB$B_XG_XMTCNT(R5)         ; Clear the no xmt's outstnd'n
              00EC C5   00EC C5 9E  1132  3655          MOVAB   UCB$Q_XG_RCV_INPR(R5),UCB$Q_XG_RCV_INPR(R5) ; Rcv inpr list
                                    1139  3656 ;
                                    1139  3657 ; Reinitialize the queue headers is case BISYNC protocol was run last
                                    1139  3658 ; in which case these fields will no longer look like queues.
                                    1139  3659 ;
              00EC C5   00EC C5 9E  1139  3660          MOVAB   UCB$Q_XG_RCV_INPR(R5),UCB$Q_XG_RCV_INPR(R5) ; Rcv inpr list
              00F0 C5   00EC C5 9E  1140  3661          MOVAB   UCB$Q_XG_RCV_INPR(R5),UCB$Q_XG_RCV_INPR+4(R5)
                           02F6 30  1147  3662          BSBW    SET_CHAR                    ; Set protocol char
                     54    24 A5 D0  114A  3663          MOVL    UCB$L_CRB(R5),R4            ; Get the CRB address
```

```
                54   2C B4    D0  114E  3664          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4    ; Get the CSR address
                             1152  3665          DSBINT  UCB$B_DIPL(R5)                  ; Lock out interupt from device
                      0268   30  1159  3666          BSBW    SET_UNIT_DDCMP                  ; Set char for DDCMP
                             115C  3667          SETIPL  #IPL$_POWER                     ; Interlock powerfail
            05 64 A5    05   E1  115F  3668          BBC     #UCB$V_POWER,UCB$W_STS(R5),10$  ; BR if no power failed
                             1164  3669          ENBINT                                  ; Enable intterrupts at FIPL
                      4C   11  1167  3670          BRB     START_POWERFAIL                 ; Branch to report powerfail
                0800 8F    A8  1169  3671  10$:    BISW    #XM$M_STS_ACTIVE,-
                  44 A5        116D  3672                  UCB$L_DEVDEPEND(R5)             ; Set active state
                             116F  3673          ENBINT                                  ; Return to fork IPL
                56   04    9A  1172  3674          MOVZBL  #DLK$C_USRINT,R6                ; Set to start the protocol
                     00    E1  1175  3675          BBC     #XM$V_CHR_MOP,-                 ; If BC then don't enter
                06 44 A5        1177  3676                  UCB$L_DEVDEPEND(R5),15$         ; maint mode
                57   04    9A  117A  3677          MOVZBL  #DLK$M_MAINT,R7                 ; Set to start maint mode
                   0003    31  117D  3678          BRW     20$
                57   01    9A  1180  3679  15$:    MOVZBL  #DLK$M_START,R7                 ; Else simply start the prot
                     28    BB  1183  3680  20$:    PUSHR   #^M<R3,R5>                      ; Save registers
             55   00D0 C5    D0  1185  3681          MOVL    UCB$A_XG_PRO_BUFFER(R5),R5      ; Get addr of start of TFB
                     58    7C  118A  3682          CLRQ    R8
                   EE71'   30  118C  3683          BSBW    DDCMP                           ; Jump to the protocol
                     28    BA  118F  3684          POPR    #^M<R3,R5>                      ; Restore registers
                56   09    91  1191  3685          CMPB    #DLK$C_ACTNOTCOM,R6             ; If EQL then protocol not
                     13    13  1194  3686          BEQL    25$                             ; in halted state don't reinit
                   0486    30  1196  3687          BSBW    FILLFREELIST                    ; Fill rcv free buffer Q
                  1E 50    E9  1199  3688          BLBC    R0,START_CIRCUIT_ERROR          ; Branch to shutdown the device
                     51    D4  119C  3689          CLRL    R1                              ; Set no status for start transmit
                   FB08    30  119E  3690          BSBW    START_TRANSMIT                  ; Branch to send STRT message
                16 50    E9  11A1  3691          BLBC    R0,START_CIRCUIT_ERROR          ; Branch to shutdown the device
                50   01    3C  11A4  3692          MOVZWL  S^#SS$_NORMAL,R0                ; Set normal return
                     19    11  11A7  3693          BRB     START_CIRCUIT_COMP              ; Comp the request
                             11A9  3694
            50   02C4 8F    3C  11A9  3695  25$:    MOVZWL  #SS$_DEVACTIVE,R0               ; Set reason for abort
                             11AE  3696          SETBIT  #XM$V_ERR_TRIB,-                ; Set to restart ciruit
                             11AE  3697                  UCB$L_DEVDEPEND(R5)
                     05    11  11B3  3698          BRB     START_CIRCUIT_ERROR
                             11B5  3699  START_POWERFAIL:
            50   0364 8F    3C  11B5  3700          MOVZWL  #SS$_POWERFAIL,R0               ; Set reason to abort
                             11BA  3701  START_CIRCUIT_ERROR:
                     50    DD  11BA  3702          PUSHL   R0                              ; Save reason to abort
                   1017    30  11BC  3703          BSBW    SHUTDOWN_CIRCUIT                ; Shutdown the circuit
                     50 8ED0  11BF  3704          POPL    R0                              ; Restore reason for abort
                             11C2  3705  START_CIRCUIT_COMP:
              53   58 A5    D0  11C2  3706          MOVL    UCB$L_IRP(R5),R3                ; Retreive packet address
              7C A5    50    B0  11C6  3707          MOVW    R0,UCB$W_BOFF(R5)               ; Set success
                       05    11  11CA  3708          RSB                                     ; Jump to complete the request
                             11CB  3709
```

```
                        11CB  3711          .SBTTL  Unit access routines
                        11CB  3712 ;++
                        11CB  3713 ;AWAIT_UNIT - Wait for unit master reset
                        11CB  3714 ;
                        11CB  3715 ; This routine uses the WFIKPCH macro to wait for the device to be master
                        11CB  3716 ; cleared. The interrupt should never occur  only the timeout.
                        11CB  3717 ;
                        11CB  3718 ; Waiting for the device to clear in this manner is a nonstandard use of
                        11CB  3719 ; the WFIPKCH macro. There are some things to note when we continue execution
                        11CB  3720 ; after the timeout has occured. First we come into the routine at DIPL instead
                        11CB  3721 ; of FIPL. Since most of the work we have to do after the timeout has
                        11CB  3722 ; happened is a FIPL then we do a SETIPL to the driver's fork ipl. We can do
                        11CB  3723 ; this because we have put in numereous checks int UNIT_INIT to make sure that
                        11CB  3724 ; IPL$_SYNCH, IPL$_TIMER and the drivers FIPL are the same. Please take all
                        11CB  3725 ; these things into consideration when changing this code.
                        11CB  3726 ;
                        11CB  3727 ;       INPUTS:   R5 = UCB address
                        11CB  3728 ;
                        11CB  3729 ;       OUTPUTS:  R0 contains the status of wait
                        11CB  3730 ;
                        11CB  3731 ;--
                        11CB  3732 AWAIT_UNIT:                                  ; Wait for unit only
                        11CB  3733          DSBINT  UCB$B_DIPL(R5)              ; Lock out interrupt from dev
                        11D2  3734
              0080 8F   A8  11D2  3735          BISW    #XG$M_MASTER_RESET,-        ; Master clear the unit
                04 A4       11D6  3736                  XG$C_MISC_REG(R4)
                        11D8  3737
                        11D8  3738          WFIKPCH AWAIT_CONT,#2               ; Wait for two seconds for
                        11E2  3739                                              ; master clear
                        11E2  3740 AWAIT_CONT:
                        11E2  3741          CLRBIT  #UCB$V_TIMOUT,UCB$W_STS(R5) ; Clear timeout
           50   0054 8F   3C  11E7  3742          MOVZWL  #SS$_CTRLERR,R0             ; Assume failure
     04 A4      0080 8F   B3  11EC  3743          BITW    #XG$M_MASTER_RESET,XG$C_MISC_REG(R4) ; If NEQL then error
                     06   13  11F2  3744          BEQL    1$
                        11F4  3745          SETIPL  #IPL$_SYNCH                 ; Return to the drivers fork
                     008F   31  11F7  3746          BRW     10$
                        11FA  3747 1$:      SETIPL  #IPL$_SYNCH                 ; Return to the drivers fork
                        11FD  3748          DSBINT  #IPL$_POWER
        03 64 A5      05   E1  1203  3749          BBC     #UCB$V_POWER,UCB$W_STS(R5),3$   ; If BC then NO power failure
                   0076   31  1208  3750          BRW     5$
                        120B  3751 3$:      ENBINT                              ; Set to DIPL
           50   0167 C5   9A  120E  3752          MOVZBL  UCB$B_XG_PROTYPE(R5),R0     ; Get protocol type
                        1213  3753          CASE    R0,TYPE=B,<-                ; Case on protocol type
                        1213  3754                  START_UNIT_LAPB,-           ; LAPB
                        1213  3755                  20$,-                       ; BISYNC
                        1213  3756                  >                           ; Fall thru on DDCMP
                        121B  3757 ;
                        121B  3758 ; Set up DDCMP defaults and other information necessary for the prptocol
                        121B  3759 ;
                03C0 8F   BB  121B  3760          PUSHR   #^M<R6,R7,R8,R9>
                     28   BB  121F  3761          PUSHR   #^M<R3,R5>
           58   0134 C5   9E  1221  3762          MOVAB   UCB$Z_XG_DDCMP(R5),R8       ; Set addr of buf with param
           59   016C C5   9E  1226  3763          MOVAB   UCB$Z_XG_DLA_ADDR(R5),R9    ; Set addr of buf with address
           50   00FC C5   9E  122B  3764          MOVAB   UCB$Z_XG_XMT_INPR(R5),R0    ; Set up address
                69   50   D0  1230  3765          MOVL    R0,DLA$A_XMT_INPR(R9)
           50   00F4 C5   9E  1233  3766          MOVAB   UCB$Q_XG_POST(R5),R0
             04 A9   50   D0  1238  3767          MOVL    R0,DLA$A_POSTQ(R9)
```

```
                55   00D0 C5    D0  123C  3768          MOVL    UCB$A_XG_PRO_BUFFER(R5),R5      ; Get protocol buffer
                     56   05    9A  1241  3769          MOVZBL  #DLK$C_CHAR,R6                  ; Set DDCMP default char
                          57    D4  1244  3770          CLRL    R7                              ; Zero the register
                57   09F0 8F    A8  1246  3771          BISW    #<DLK$M_MSGCNT!-                ; indicate chara to set
                                       124B  3772                  DLK$M_SELTIM!-
                                       124B  3773                  DLK$M_SELWAIT!-
                                       124B  3774                  DLK$M_REPTIM!-
                                       124B  3775                  DLK$M_REPWAIT!-
                                       124B  3776                  DLK$M_SETDEF>,R7
                       EDB2'   30  124B  3777          BSBW    DDCMP
                          28   BA  124E  3778          POPR    #^M<R3,R5>
                                       1250  3779 ;
                                       1250  3780 ; Before starting up the DDCMP timer make sure that the driver can not
                                       1250  3781 ; be reloaded while the timer is running.
                                       1250  3782 ;
00000000D'EF         04    88  1250  3783          BISB    #DPT$M_NOUNLOAD,DPT$TAB+DPT$B_FLAGS
                                       1257  3784 ;
                                       1257  3785 ; Start up the DDCMP timer. First set up registers with params for call
                                       1257  3786 ; then make the call to the protocol.
                                       1257  3787 ;
                          28   BB  1257  3788          PUSHR   #^M<R3,R5>
                     56   06    9A  1259  3789          MOVZBL  #DLK$C_START_TIMER,R6           ; Set up R6 with the DDCMP comm
                57   1D77'CF    DE  125C  3790          MOVAL   W^DEVTIMER,R7                   ; R7 must have device timer addr
                     58   55    D0  1261  3791          MOVL    R5,R8                           ; R8 must have UCB addr
                          59    D4  1264  3792          CLRL    R9                              ; Clear R9
                55   00D0 C5    D0  1266  3793          MOVL    UCB$A_XG_PRO_BUFFER(R5),R5      ; Get addr of start of TFB
                       ED92'   30  126B  3794          BSBW    DDCMP                           ; Call protocol
                          28   BA  126E  3795          POPR    #^M<R3,R5>
                     03C0 8F    BA  1270  3796          POPR    #^M<R6,R7,R8,R9>
                     50   01    3C  1274  3797          MOVZWL  S^#SS$_NORMAL,R0                ; Set success
                  51   44 A5    D0  1277  3798          MOVL    UCB$L_DEVDEPEND(R5),R1          ; Set device charateristics
                                       127B  3799          REQCOM                          ; Complete the request
                                       1281  3800
                50   0364 8F    3C  1281  3801 5$:      MOVZWL  #SS$_POWERFAIL,R0               ; Treat as failure to init
                                       1286  3802          ENBINT                          ; Set to FIPL
                          50   DD  1289  3803 10$:     PUSHL   R0                              ; Save abort reason
                        0E70   30  128B  3804          BSBW    SHUTDOWN_LINE                   ; Shut down the device
                          50 8ED0  128E  3805          POPL    R0                              ; Restore abort reason
                  51   44 A5    D0  1291  3806          MOVL    UCB$L_DEVDEPEND(R5),R1          ; Set device charateristics
                                       1295  3807          REQCOM                          ; Complete the request
                        01F0   31  1298  3808 20$:     BRW     START_UNIT_BISYNC
                                       129E  3809
```

```
                                129E  3811 ;++
                                129E  3812 ; START_UNIT_LAPB - Start line and set char for LAPB
                                129E  3813 ;
                                129E  3814 ; Lapb does not have a seperate start up for line and circuit, therefore
                                129E  3815 ; when a start up on the line is received then it must also start
                                129E  3816 ; up the circuit. This includes setting up the proper fields in the
                                129E  3817 ; UCB and setting the board with the proper fields
                                129E  3818 ;
                                129E  3819 ;       INPUT   R3 = IRP address
                                129E  3820 ;               R5 = UCB address
                                129E  3821 ;--
                                129E  3822 START_UNIT_LAPB:
                          0B    E1  129E  3823         BBC     #XM$V_STS_ACTIVE,-              ; If BC then device and
                    08 44 A5        12A0  3824                 UCB$L_DEVDEPEND(R5),5$          ; protocol are not active
                50   02C4 8F    3C  12A3  3825         MOVZWL  #SS$_DEVACTIVE,R0               ; Return error
                       00FF     31  12A8  3826         BRW     SET_UNIT_LAPB_ERROR
                                    12AB  3827
                          02    E0  12AB  3828 5$:     BBS     #XG_DS_V_INITED,-               ; If BS then line started
                    08 68 A5        12AD  3829                 UCB$W_DEVSTS(R5),8$
                50   20D4 8F    3C  12B0  3830         MOVZWL  #SS$_DEVINACT,R0                ; Else return device inactive
                       00F2     31  12B5  3831         BRW     SET_UNIT_LAPB_ERROR
                                    12B8  3832
                          AA    BC  12B8  3833 8$:     BICW    #^C<XG_DS_M_INITED!-            ; Reset bit fields
                                12B9  3834                 XG_DS_M_CTSTOE_RUN>,-
                68 A5   BFFB 8F     12B9  3835                 UCB$W_DEVSTS(R5)
            44 A5   FFFFF700 8F CA  12BE  3836         BICL    #XG$C_CIR_PAR,UCB$L_DEVDEPEND(R5) ; Reset all Read/Write flags
                      0120 C5   94  12C6  3837         CLRB    UCB$B_XG_XSTATE(R5)             ; Set XMTer state to idle
                      0164 C5   94  12CA  3838         CLRB    UCB$B_XG_XMTCNT(R5)             ; Clear the no xmt's outstnd'n
                      0165 C5   B4  12CE  3839         CLRW    UCB$W_XG_MFDLEN(R5)             ; Set zero protocol header
                                    12D2  3840 ;
                                    12D2  3841 ; Reinitialize the queue headers is case BISYNC protocol was run last
                                    12D2  3842 ; in which case these fields will no longer look like queues.
                                    12D2  3843 ;
        00EC C5   00EC C5   9E      12D2  3844         MOVAB   UCB$Q_XG_RCV_INPR(R5),UCB$Q_XG_RCV_INPR(R5) ; Rcv inpr list
        00F0 C5   00EC C5   9E      12D9  3845         MOVAB   UCB$Q_XG_RCV_INPR(R5),UCB$Q_XG_RCV_INPR+4(R5)
             54   00D0 C5   D0      12E0  3846         MOVL    UCB$A_XG_PRO_BUFFER(R5),R4      ; Set protocol buffer
                                    12E5  3847
                                    12E5  3848         ASSUME  LAPB$W_DEIBC EQ LAPB$W_DEITYP+2
                                    12E5  3849
          12 A4    03FC BF    3C  12E5  3850         MOVZWL  #NMA$C_CTCIR_DEI,LAPB$W_DEITYP(R4) ; Set kind of error
                      16 A4   94  12EB  3851         CLRB    LAPB$B_DEI(R4)                  ; Clear the field
                      10 A4   94  12EE  3852         CLRB    LAPB$B_XQCNT(R4)                ; Clear the xmt cnt field
             20 A4    20 A4   9E  12F1  3853         MOVAB   LAPB$Q_XMTQ(R4),LAPB$Q_XMTQ(R4) ; Initialize queues
             24 A4    20 A4   9E  12F6  3854         MOVAB   LAPB$Q_XMTQ(R4),LAPB$Q_XMTQ+4(R4)
             28 A4    28 A4   9E  12FB  3855         MOVAB   LAPB$Q_CLEANQ(R4),LAPB$Q_CLEANQ(R4) ; Initialize queues
             2C A4    28 A4   9E  1300  3856         MOVAB   LAPB$Q_CLEANQ(R4),LAPB$Q_CLEANQ+4(R4)
             30 A4    30 A4   9E  1305  3857         MOVAB   LAPB$Q_BLANK(R4),LAPB$Q_BLANK(R4) ; Initialize queues
             34 A4    30 A4   9E  130A  3858         MOVAB   LAPB$Q_BLANK(R4),LAPB$Q_BLANK+4(R4) ; Initialize queues
                54    24 A5   D0  130F  3859         MOVL    UCB$L_CRB(R5),R4                ; Get the CRB address
                54    2C B4   D0  1313  3860         MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4    ; Get the CSR address
                                    1317  3861         DSBINT  UCB$B_DIPL(R5)                  ; Lock out interupt from device
              52    0144 C5   DE  131E  3862         MOVAL   UCB$Z_XG_SYNC(R5),R2            ; Get the addr of SYNC params
                04 A4    00   B0  1323  3863         MOVW    #0,XG$C_MISC_REG(R4)            ; Set the first Ind reg
                   50    62   9B  1327  3864         MOVZBW  XG$B_ERR_CNTRL(R2),R0           ; Set type of error checking
                03    01 A2   F0  132A  3865         INSV    XG$B_PROTOCOL(R2),#XG$V_PROTOCOL,- ; Set type of protocol
                   50    03       132E  3866                 #XG$S_PROTOCOL,R0
                06 A4    50   B0  1330  3867         MOVW    R0,XG$C_IND_ADDR(R4)            ; Init IR0
```

```
                06 A4    00    B0  1334  3868          MOVW    #0,XG$C_IND_ADDR(R4)            ; Init IR1
                   50    B4    1338  3869          CLRW    R0
                08   04 A2    F0  133A  3870          INSV    XG$B_BAUD(R2),#XG$V_XMT_BRG,-   ; Set baud rate
                50    04          133E  3871                  #XG$S_XMT_BRG,R0
                06 A4    50    B0  1340  3872          MOVW    R0,XG$C_IND_ADDR(R4)            ; Init IR2
                06 A4    00    B0  1344  3873          MOVW    #0,XG$C_IND_ADDR(R4)            ; Init IR3
                06 A4    00    B0  1348  3874          MOVW    #0,XG$C_IND_ADDR(R4)            ; Init IR4
                06 A4    00    B0  134C  3875          MOVW    #0,XG$C_IND_ADDR(R4)            ; Init IR5
                                   1350  3876
                                   1350  3877 ; Set the receive and transmit csr's
                                   1350  3878
                   50    20    9B  1350  3879          MOVZBW  #XG$M_INT_ENABLE,R0             ; Set to enable interrupts
                                   1353  3880          ASSUME  NMA$C_LINCN_NOR EQ 0
                 0152 C5    95  1353  3881          TSTB    UCB$B_XG_CON(R5)                ; Normal controller?
                      05    13  1357  3882          BEQL    20$                             ; If EQL then yes
             50   0100 8F    A8  1359  3883          BISW2   #XG$M_ILP_XCS,R0                ; Set internal clock src
                                   135E  3884                                                  ; and internal loopback
                   51    50    D0  135E  3885 20$:    MOVL    R0,R1                           ; Move into R1
                 0157 C5    F0  1361  3886          INSV    UCB$B_XG_MNT_LOOPB(R5),-        ; Set LOOPB type
             51    02    09          1365  3887                  #XG$V_LOOP_TYPE,#XG$S_LOOP_TYPE,R1
                   64    51    B0  1368  3888          MOVW    R1,XG$C_RCV_CSR(R4)             ; Set for RCV's
                   50    08    A8  136B  3889          BISW2   #XG$M_TERM_IDL,R0               ; Set XMTer to IDLE
                02 A4    50    B0  136E  3890          MOVW    R0,XG$C_XMT_CSR(R4)             ; Set for XMT's
             50   0200 8F    3C  1372  3891          MOVZWL  #XG$M_DTR,R0                    ; Set DTR on
             50   1000 8F    A8  1377  3892          BISW    #XG$M_RTS,R0                    ; Set RTS on
                04 A4    04    B0  137C  3893          MOVW    #XG$C_MODEM,XG$C_MISC_REG(R4)
                06 A4    50    B0  1380  3894          MOVW    R0,XG$C_IND_ADDR(R4)
                                   1384  3895          SETIPL  #IPL$_POWER                     ; Interlock powerfail
          0A 64 A5    05    E1  1387  3896          BBC     #UCB$V_POWER,UCB$W_STS(R5),10$  ; BR if no power failed
                                   138C  3897          ENBINT                                  ; Enable intterrupts at FIPL
             50   0364 8F    3C  138F  3898          MOVZWL  #SS$_POWERFAIL,R0               ; Set reason to abort
                      14    11  1394  3899          BRB     SET_UNIT_LAPB_ERROR
                 0800 8F    A8  1396  3900 10$:    BISW    #XM$M_STS_ACTIVE,-
                   44 A5          139A  3901                  UCB$L_DEVDEPEND(R5)             ; Set active state
                                   139C  3902          ENBINT                                  ; Return to fork IPL
                     027D    30  139F  3903          BSBW    FILLFREELIST                    ; Fill rcv free buffer Q
                   05 50     E9  13A2  3904          BLBC    R0,SET_UNIT_LAPB_ERROR          ; Branch to shutdown the device
                50    01    3C  13A5  3905          MOVZWL  S^#SS$_NORMAL,R0                ; Set normal return
                      08    11  13A8  3906          BRB     SET_UNIT_LAPB_COMP              ; Comp the request
                                   13AA  3907
                                   13AA  3908 SET_UNIT_LAPB_ERROR:
                      50    DD  13AA  3909          PUSHL   R0                              ; Save reason to abort
                   0D4F    30  13AC  3910          BSBW    SHUTDOWN_LINE                   ; Shutdown the line
                      50 8ED0  13AF  3911          POPL    R0                              ; Restore reason for abort
                                   13B2  3912 SET_UNIT_LAPB_COMP:
             53    58 A5    D0  13B2  3913          MOVL    UCB$L_IRP(R5),R3                ; Retreive packet address
          7C A5    50    B0  13B6  3914          MOVW    R0,UCB$W_BOFF(R5)               ; Set success
             51    44 A5    D0  13BA  3915          MOVL    UCB$L_DEVDEPEND(R5),R1          ; Set device depend char
                                   13BE  3916          REQCOM                                  ; Jump to complete the request
                                   13C4  3917
```

```
                        13C4  3919 ;++
                        13C4  3920 ;SET_UNIT_DDCMP - Set default characters on board protocol support is DDCMP
                        13C4  3921 ;
                        13C4  3922 ; Sets the device characteristics to be that for DDCMP
                        13C4  3923 ;
                        13C4  3924 ; The defaults are as follows
                        13C4  3925 ;
                        13C4  3926 ;       IR0:    Proctol = DDCMP, Error = CRC16; Bits/char = 8
                        13C4  3927 ;       IR1:    Clear RCV error
                        13C4  3928 ;       IR2:    Clear XMT error, Set default baud rate = 19.2kb/sec
                        13C4  3929 ;       IR3:    Set number of syncs = 8  Set sync char = 150
                        13C4  3930 ;       IR4:    Clear data set change bits
                        13C4  3931 ;       IR5:    Set station address
                        13C4  3932 ;
                        13C4  3933 ;       INPUTS: R4 = CSR address
                        13C4  3934 ;               R5 = UCB address
                        13C4  3935 ;
                        13C4  3936 ;--
                        13C4  3937 SET_UNIT_DDCMP:
          52  0144 C5  DE  13C4  3938         MOVAL   UCB$Z_XG_SYNC(R5),R2            ; Get the addr of SYNC params
            04 A4  00  B0  13C9  3939         MOVW    #0,XG$C_MISC_REG(R4)            ; Set the first Ind reg
               50  62  9B  13CD  3940         MOVZBW  XG$B_ERR_CNTRL(R2),R0           ; Set type of error checking
            50  40 8F  88  13D0  3941         BISB2   #XG$M_STRIP_SYNC,R0             ; Set to strip excess sync char
            06 A4  50  B0  13D4  3942         MOVW    R0,XG$C_IND_ADDR(R4)            ; Init IR0
            06 A4  00  B0  13D8  3943         MOVW    #0,XG$C_IND_ADDR(R4)            ; Init IR1
                   50  B4  13DC  3944         CLRW    R0
            08  04 A2  F0  13DE  3945         INSV    XG$B_BAUD(R2),#XG$V_XMT_BRG,-   ; Set baud rate
               50  04        13E2  3946                 #XG$S_XMT_BRG,R0
            06 A4  50  B0  13E4  3947         MOVW    R0,XG$C_IND_ADDR(R4)            ; Init IR2
            50  05 A2  9B  13E8  3948         MOVZBW  XG$B_NUM_SYNC(R2),R0            ; Set number of syncs
            08  06 A2  F0  13EC  3949         INSV    XG$B_SYNC_REG(R2),#XG$V_SYNC,-  ; Set sync char
               50  08        13F0  3950                 #XG$S_SYNC,R0
            06 A4  50  B0  13F2  3951         MOVW    R0,XG$C_IND_ADDR(R4)            ; Init IR3
            06 A4  00  B0  13F6  3952         MOVW    #0,XG$C_IND_ADDR(R4)            ; Init IR4
          50  00D0 C5  D0  13FA  3953         MOVL    UCB$A_XG_PRO_BUFFER(R5),R0      ; Get protocol buffer type
            50  0D A0  9B  13FF  3954         MOVZBW  TF$B_ADDR(R0),R0                ; Get trib address into word
            06 A4  50  B0  1403  3955         MOVW    R0,XG$C_IND_ADDR(R4)            ; Init IR5
                           1407  3956
                           1407  3957 ; Set the receive and transmit csr's
                           1407  3958
               50  20  9B  1407  3959         MOVZBW  #XG$M_INT_ENABLE,R0             ; Set to enable interrupts
                           140A  3960         ASSUME  NMA$C_LINCN_NOR EQ 0
               0152 C5 95  140A  3961         TSTB    UCB$B_XG_CON(R5)                ; Normal controller?
                   05  13  140E  3962         BEQL    20$                             ; If EQL then yes
          50   0100 8F  A8  1410  3963         BISW2   #XG$M_ILP_XCS,R0                ; Set internal clock src
                           1415  3964                                                 ; and internal loopback
               51  50  D0  1415  3965 20$:     MOVL    R0,R1                           ; Move into R1
               0157 C5 F0  1418  3966         INSV    UCB$B_XG_MNT_LOOPB(R5),-        ; Set LOOPB type
          51   02  09        141C  3967                 #XG$V_LOOP_TYPE,#XG$S_LOOP_TYPE,R1
                   02  91  141F  3968         CMPB    #NMA$C_LINPR_TRI,-              ; If trib then set bit on board
               0150 C5        1421  3969                 UCB$B_XG_PRO(R5)                ; so that address is checked
                   05  12  1424  3970         BNEQ    30$
          51   2000 8F  A8  1426  3971         BISW    #XG$M_PRI_SEC_STN,R1            ; Set that this is a trib
               64  51  B0  142B  3972 30$:     MOVW    R1,XG$C_RCV_CSR(R4)             ; Set for RCV's
            02 A4  50  B0  142E  3973         MOVW    R0,XG$C_XMT_CSR(R4)             ; Set for XMT's
          50   0200 8F  3C  1432  3974         MOVZWL  #XG$M_DTR,R0                    ; Set DTR on
            04 A4  04  B0  1437  3975         MOVW    #XG$C_MODEM,XG$C_MISC_REG(R4)
```

```
                06 A4    50    B0  143B  3976          MOVW    R0,XG$C_IND_ADDR(R4)
                               05  143F  3977          RSB
                                   1440  3978
                                   1440  3979
                                   1440  3980
```

```
                                1440  3982          .SBTTL  Set protocol characteristic
                                1440  3983 ;++
                                1440  3984 ; SET_CHAR - Set characteristics of protocol
                                1440  3985 ;
                                1440  3986 ; FUNCTIONAL DESCRIPTION:
                                1440  3987 ;
                                1440  3988 ; This routine is entered to set characteristic on a halted protocol.
                                1440  3989 ;
                                1440  3990 ; INPUTS:
                                1440  3991 ;
                                1440  3992 ;       R5 = UCB address
                                1440  3993 ;
                                1440  3994 ;
                                1440  3995 ; OUTPUTS:
                                1440  3996 ;
                                1440  3997 ;       New protocol characteristics given to protocol.
                                1440  3998 ;--
                                1440  3999 SET_CHAR:
                          57    D4    1440  4000          CLRL    R7                              ; Set no charac to change
                0165 C5   06    B0    1442  4001          MOVW    #MFD$K_LENGTH,UCB$W_XG_MFDLEN(R5) ; Set protocol header sz
           58   0134 C5          9E    1447  4002          MOVAB   UCB$Z_XG_DDCMP(R5),R8           ; Get trib parameter block
                  01 A8    90    144C  4003          MOVB    DLK$B_MSGCNT(R8),-              ; Use this field for RTO's
                  02 A8          144F  4004                  DLK$B_MAXREP(R8)
                  01 A8    90    1451  4005          MOVB    DLK$B_MSGCNT(R8),-              ; And for number of Select
                  03 A8          1454  4006                  DLK$B_MAXSEL(R8)                ; intervals
                  04 A8    B0    1456  4007          MOVW    DLK$W_REPWAIT(R8),-             ; Use repwait for selwait
                  06 A8          1459  4008                  DLK$W_SELWAIT(R8)
           57   01F0 8F    A8    145B  4009          BISW    #<DLK$M_MSGCNT!-                ; Set to reset these param
                                1460  4010                  DLK$M_SELTIM!-
                                1460  4011                  DLK$M_SELWAIT!-
                                1460  4012                  DLK$M_REPTIM!-
                                1460  4013                  DLK$M_REPWAIT>,R7
                                1460  4014
                                1460  4015          ASSUME  NMA$C_LINPR_POI EQ 0
                                1460  4016          SETBIT  #DLK$V_SETDEF,R7                ; Set default char
                0150 C5   95    1464  4017          TSTB    UCB$B_XG_PRO(R5)                ; If EQL then point to point
                      08  13    1468  4018          BEQL    10$
                                146A  4019          CLRBIT  #DLK$V_SETDEF,R7                ; Use user given char
                                146E  4020          SETBIT  #DLK$V_STATYP,R7                ; Else set trib
                                1472  4021 10$:     ASSUME  NMA$C_DPX_FUL EQ 0
                0151 C5   95    1472  4022          TSTB    UCB$B_XG_DUP(R5)                ; If EQL the full duplex
                      04  13    1476  4023          BEQL    20$
                                1478  4024          SETBIT  #DLK$V_DUPLEX,R7                ; Else set half duplex
                      28  BB    147C  4025 20$:     PUSHR   #^M<R3,R5>                      ; Save registers
           55   00D0 C5   D0    147E  4026          MOVL    UCB$A_XG_PRO_BUFFER(R5),R5      ; Get addr of start of TFB
                56   05   9A    1483  4027          MOVZBL  #DLK$C_CHAR,R6                  ; Set charac in DDCMP
                     59   D4    1486  4028          CLRL    R9
                        EB75'   30    1488  4029          BSBW    DDCMP
                      28  BA    148B  4030          POPR    #^M<R3,R5>                      ; Restore the registers
                          05    148D  4031 30$:     RSB
                                148E  4032
```

```
                                148E  4034          .SBTTL  START_UNIT_BISYNC - Start up device in BISYNC mode
                                148E  4035 ;++
                                148E  4036 ; START_UNIT_BISYNC - Start up device in Bisync mode
                                148E  4037 ;
                                148E  4038 ; As in LAPB, Bisync does not have a seperate start up for the line and circuit,
                                148E  4039 ; therefore when issuing a startup on the line, we must start the entire
                                148E  4040 ; protocol. It must also set the necessary field both in the UCB and on the
                                148E  4041 ; device.
                                148E  4042 ;
                                148E  4043 ;       INPUTS:
                                148E  4044 ;               R5 = Address of UCB
                                148E  4045 ;
                                148E  4046 ;       OUTPUTS
                                148E  4047 ;               R0 = Status of request
                                148E  4048 ;               R1 = Contents of UCB$L_DEVDEPEND
                                148E  4049 ;               R3 = IRP address
                                148E  4050 ;               R5 = UCB address
                                148E  4051 ;--
                                148E  4052 START_UNIT_BISYNC:
            08 44 A5    0B  E1  148E  4053          BBC     #XM$V_STS_ACTIVE,UCB$L_DEVDEPEND(R5),10$ ; If BC device not active
             50    02C4 8F  3C  1493  4054          MOVZWL  #SS$_DEVACTIVE,R0               ; Set device active
                   0114     31  1498  4055          BRW     START_UNIT_BISYNC_ERROR         ; Complete in error
                            149B  4056
            08 68 A5    02  E0  149B  4057 10$:     BBS     #XG_DS_V_INITED,UCB$W_DEVSTS(R5),20$ ; If BS then line started
             50    02C4 8F  3C  14A0  4058          MOVZWL  #SS$_DEVACTIVE,R0               ; Else complete request
                   0107     31  14A5  4059          BRW     START_UNIT_BISYNC_ERROR         ; in error
                            14A8  4060
                            14A8  4061 20$:     .IF     DF BISYNC$$$
                            14A8  4062          moval   w^bisync_framing_routine,ucb$a_xg_frame_addr(r5) ; For testing BISYN
                            14A8  4063          .ENDC   ;DF BISYNC$$$
                            14A8  4064
        68 A5    BFFB 8F    AA  14A8  4065          BICW    #^C<XG_DS_M_INITED!-            ; Reset flags
                                14AE  4066                      XG_DS_M_CTSTQE_RUN>,UCB$W_DEVSTS(R5)
    44 A5    FFFFF700 8F    CA  14AE  4067          BICL    #XG$C_CIR_PAR,UCB$L_DEVDEPEND(R5) ; Reset read/write flags
                0120 C5    94  14B6  4068          CLRB    UCB$B_XG_XSTATE(R5)             ; Set XMTer state to idle
                0164 C5    94  14BA  4069          CLRB    UCB$B_XG_XMTCNT(R5)             ; Clear XMT's outstanding
                0165 C5    B4  14BE  4070          CLRW    UCB$W_XG_MFDLEN(R5)             ; Set zero protocol header
          54    00D0 C5    D0  14C2  4071          MOVL    UCB$A_XG_PRO_BUFFER(R5),R4      ; Get protocol buffer address
       30 A4      30 A4    9E  14C7  4072          MOVAB   BISYNC$Q_XMTQ(R4),BISYNC$Q_XMTQ(R4) ; Initialize queues
       34 A4      30 A4    9E  14CC  4073          MOVAB   BISYNC$Q_XMTQ(R4),BISYNC$Q_XMTQ+4(R4)
       28 A4      28 A4    9E  14D1  4074          MOVAB   BISYNC$Q_CLEANQ(R4),BISYNC$Q_CLEANQ(R4) ; Initialize queues
       2C A4      28 A4    9E  14D6  4075          MOVAB   BISYNC$Q_CLEANQ(R4),BISYNC$Q_CLEANQ+4(R4)
       20 A4      20 A4    9E  14DB  4076          MOVAB   BISYNC$Q_BLANK(R4),BISYNC$Q_BLANK(R4) ; Initialize queues
       24 A4      20 A4    9E  14E0  4077          MOVAB   BISYNC$Q_BLANK(R4),BISYNC$Q_BLANK+4(R4) ; Initialize queues
          54      24 A5    D0  14E5  4078          MOVL    UCB$L_CRB(R5),R4                ; Get the CRB address
          54      2C B4    D0  14E9  4079          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4    ; Get the CSR
                              14ED  4080          DSBINT  UCB$B_DIPL(R5)                  ; Disable interrupts
          52    0144 C5    DE  14F4  4081          MOVAL   UCB$Z_XG_SYNC(R5),R2            ; Get the addr of sync params
            04 A4    00    B0  14F9  4082          MOVW    #0,XG$C_MISC_REG(R4)            ; Set the first indirect reg
                50    62    9B  14FD  4083          MOVZBW  XG$B_ERR_CNTRL(R2),R0           ; Set type of error control
            03    01 A2    F0  1500  4084          INSV    XG$B_PROTOCOL(R2),#XG$V_PROTOCOL,- ; Set type of protocol
                50    03        1504  4085                  #XG$S_PROTOCOL,R0
                              1506  4086          SETBIT  #XG$V_STRIP_SYNC,R0             ; Set to strip sync characters
            06 A4    50    B0  150A  4087          MOVW    R0,XG$C_IND_ADDR(R4)            ; Set IR0
            06 A4    00    B0  150E  4088          MOVW    #0,XG$C_IND_ADDR(R4)            ; Set IR1
                50    D4  1512  4089          CLRL    R0
            08    04 A2    F0  1514  4090          INSV    XG$B_BAUD(R2),#XG$V_XMT_BRG,-   ; Set baud rate
```

```
                                                 I 9
                50   04        1518  4091                    #XG$S_XMT_BRG,R0
          06 A4 50   B0        151A  4092            MOVW    R0,XG$C_IND_ADDR(R4)            ; Set IR2
          50   05 A2 9B        151E  4093            MOVZBW  XG$B_NUM_SYNC(R2),R0            ; Set number of syncs
          08   06 A2 F0        1522  4094            INSV    XG$B_SYNC_REG(R2),#XG$V_SYNC,-  ; Set sync char
                50   08        1526  4095                    #XG$S_SYNC,R0
          06 A4 50   B0        1528  4096            MOVW    R0,XG$C_IND_ADDR(R4)            ; Init IR3
          06 A4 00   B0        152C  4097            MOVW    #0,XG$C_IND_ADDR(R4)            ; Set IR4
          06 A4 00   B0        1530  4098            MOVW    #0,XG$C_IND_ADDR(R4)            ; Set IR5
                               1534  4099 ;
                               1534  4100 ; Set the receive and tranmit register. Do not enable interrupts on the
                               1534  4101 ; receiver side.
                               1534  4102 ;
                50   D4        1534  4103            CLRL    R0
                               1536  4104
                               1536  4105            ASSUME  NMA$C_LINCN_NOR EQ 0
                               1536  4106
          0152 C5   95         1536  4107            TSTB    UCB$B_XG_CON(R5)                ; If EQL then controller not
                05   13        153A  4108            BEQL    30$                             ; loopbacked
     50   0100 8F   A8         153C  4109            BISW2   #XG$M_ILP_XCS,R0                ; Else set loopback
          51   50   D0         1541  4110 30$:      MOVL    R0,R1
          0157 C5   F0         1544  4111            INSV    UCB$B_XG_MNT_LOOPB(R5),-        ; Set loop back type
     51   02   09              1548  4112                    #XG$V_LOOP_TYPE,#XG$S_LOOP_TYPE,R1
          64   51   B0         154B  4113            MOVW    R1,XG$C_RCV_CSR(R4)             ; Set the RCV csr
          50   20   A8         154E  4114            BISW2   #XG$M_INT_ENABLE,R0             ; Set XMT char's
        02 A4  50   B0         1551  4115            MOVW    R0,XG$C_XMT_CSR(R4)             ; Set the XMT csr
     50   0200 8F   3C         1555  4116            MOVZWL  #XG$M_DTR,R0                    ; Set DTR high
        04 A4  04   B0         155A  4117            MOVW    #XG$C_MODEM,XG$C_MISC_REG(R4)   ; Set to set up the modem IR
        06 A4  50   B0         155E  4118            MOVW    R0,XG$C_IND_ADDR(R4)            ; Set up modem IR4
                               1562  4119            SETIPL  #IPL$_POWER                     ; Lock out interrupts
  0A 64 A5  05   E1            1565  4120            BBC     #UCB$V_POWER,UCB$W_STS(R5),40$  ; Check for power failure
                               156A  4121            ENBINT                                  ; Return to fork IPL
     50   0364 8F   3C         156D  4122            MOVZWL  #SS$_POWERFAIL,R0               ; Set power failed
                3B   11         1572  4123            BRB     START_UNIT_BISYNC_ERROR         ; Complete startupin error
                               1574  4124 40$:       SETBIT  #XM$V_STS_ACTIVE,UCB$L_DEVDEPEND(R5) ; Set active bit
                               1579  4125            ENBINT                                  ; Enable interrupts
     54   00D0 C5   D0         157C  4126            MOVL    UCB$A_XG_PRO_BUFFER(R5),R4      ; Get protocol buffer
  14 A4   015C C5   7D         1581  4127            MOVQ    UCB$Q_XG_STATE_INFO(R5),BISYNC$Q_INIT_STATE_INFO(R4) ; Set init stat
                               1587  4128
                               1587  4129            ASSUME  BISYNC$W_STATUS EQ BISYNC$W_RCV_INDEX+2
                               1587  4130            ASSUME  BISYNC$W_DROP_RCV EQ BISYNC$B_XQCNT+1
                               1587  4131
          0C A4  D4            1587  4132            CLRL    BISYNC$W_RCV_INDEX(R4)          ; Clear index and statsu fields
          10 A4  D4            158A  4133            CLRL    BISYNC$B_XQCNT(R4)              ; Clear xqcnt and drop rcv field
             54  DD            158D  4134            PUSHL   R4
           008D  30            158F  4135            BSBW    FILLFREELIST                    ; Fill free list with buffers
             54 8ED0           1592  4136            POPL    R4
                               1595  4137 ;
                               1595  4138 ; Take a buffer from the FREELIST to use as the receive buffer. Because of
                               1595  4139 ; the nature of how the device works in BISYNC mode, we will not use the
                               1595  4140 ; double buffer capability of the DMF32 sync port for receives. Receives
                               1595  4141 ; on this device will work on a timer base. Whenever the timer goes off we
                               1595  4142 ; will check to see if anymore characters have been input. It they have
                               1595  4143 ; then we will give them to the framming routine and depending on what the
                               1595  4144 ; framming routine decides the buffer may or may not be completed. If the
                               1595  4145 ; buffer is completed then all fields necesary for receiving are reset.
                               1595  4146 ;
     50   00E4 D5   0F         1595  4147            REMQUE  @UCB$Q_XG_FREE(R5),R0           ; Get a buffer for rv's
```

```
                 13   1D  159A  4148          BVS     START_UNIT_BISYNC_ERROR          ; If VS then no buffers alloc
1C A4            50   D0  159C  4149          MOVL    R0,BISYNC$A_RCV_BUFFER(R4)       ; Save the rcv buffer
   00EC C5       7C       15A0  4150          CLRQ    UCB$Q_XG_RCV_INPR(R5)            ; Clear no buffer rcving
                          15A4  4151 ;
                          15A4  4152 ; Now go off and start the timer to start looking for receive characters. The
                          15A4  4153 ; timer is responsible for starting up the receive requests. This means that
                          15A4  4154 ; START_RECEIVE routine is not called when running in BISYNC mode
                          15A4  4155 ;
              0022   30   15A4  4156          BSBW    START_BISYNC_TIMER               ; Start timer and rcv's
            05 50    E9   15A7  4157          BLBC    R0,START_UNIT_BISYNC_ERROR       ; If LBC then error on startup
         50    01    3C   15AA  4158          MOVZWL  S^#SS$_NORMAL,R0                 ; Else set success
               08    11   15AD  4159          BRB     START_UNIT_BISYNC_COMP
                          15AF  4160
                          15AF  4161 START_UNIT_BISYNC_ERROR:
               50   DD   15AF  4162          PUSHL   R0
            0B4A    30   15B1  4163          BSBW    SHUTDOWN_LINE
               50 8ED0   15B4  4164          POPL    R0
                          15B7  4165
                          15B7  4166 START_UNIT_BISYNC_COMP:
53      58 A5    D0   15B7  4167          MOVL    UCB$L_IRP(R5),R3                 ; Get IRP
7C A5    50    B0   15BB  4168          MOVW    R0,UCB$W_BOFF(R5)                ; Set QIO status
51    44 A5    D0   15BF  4169          MOVL    UCB$L_DEVDEPEND(R5),R1           ; Set device depend char
                    15C3  4170          REQCOM                                   ; Jump to complete the request
```

```
                                15C9  4172          .SBTTL  START_BISYNC_TIMER - Start up the BISYNC timer
                                15C9  4173 ;++
                                15C9  4174 ; START_BISYNC_TIMER - Start up the BISYNC timer
                                15C9  4175 ;
                                15C9  4176 ; This routine is called to start up the timer and the receiver on the DMF32.
                                15C9  4177 ; The reasons we can't use START_RECEIVE to start up receives are two fold.
                                15C9  4178 ; First we will never have more than one receive posted on the device at any
                                15C9  4179 ; time. Second because of the way we have to handle receive characters. That is
                                15C9  4180 ; running a timer, which when ticks looks in the receive buffer to see if any
                                15C9  4181 ; characters were input. This makes having two buffers posted rather
                                15C9  4182 ; useless.
                                15C9  4183 ;
                                15C9  4184 ;       INPUTS
                                15C9  4185 ;               R4 = Address of protocol buffer
                                15C9  4186 ;               R5 = UCB address
                                15C9  4187 ;
                                15C9  4188 ;       OUTPUT
                                15C9  4189 ;               R5 = UCB address
                                15C9  4190 ;               The receive for BISYNC is started up.
                                15C9  4191 ;
                                15C9  4192 ;--
                                15C9  4193 START_BISYNC_TIMER:
                      55    DD  15C9  4194          PUSHL   R5                              ; Save R5
                   53   55    D0  15CB  4195          MOVL    R5,R3                           ; Keep the UCB ADDRESS
                55   38 A4    9E  15CE  4196          MOVAB   BISYNC$L_TQE(R4),R5             ; Get TQE address
         0C A5   00001E18'GF    3E  15D2  4197          MOVAW   G^BISYNC_TIMER,TQE$L_FPC(R5)    ; Set address of timer routine
 20 A5   00000000 0003D090 8F    7D  15DA  4198          MOVQ    #XG$C_BISYNC_DELTA,TQE$Q_DELTA(R5) ; Set delta
                10 A5   53    D0  15E6  4199          MOVL    R3,TQE$L_FR3(R5)                ; Set TQE with UCB address
                14 A5   54    D0  15EA  4200          MOVL    R4,TQE$L_FR4(R5)                ; Set TQE with pro buffer addr
                0B A5   05    90  15EE  4201          MOVB    #TQE$C_SSREPT,TQE$B_RQTYPE(R5)  ; Set system and repeatable
                   28 A5    94  15F2  4202          CLRB    TQE$B_RMOD(R5)                  ; Set no access mode
            50  00000000'GF    7D  15F5  4203          MOVQ    G^EXE$GQ_SYSTIME,R0             ; Set system time
                00000000'GF    16  15FC  4204          JSB     G^EXE$INSTIMQ                   ; Insert on timer queue
                         55 8ED0  1602  4205          POPL    R5                              ; Reset UCB address
                                  1605  4206          SETBIT  #BISYNC$V_TIMER_RUNNING,BISYNC$W_STATUS(R4)
0000000D'EF     04    88  160A  4207          BISB    #DPT$M_NOUNLOAD,DPT$TAB+DPT$B_FLAGS ; Set driver not reloadable
                                  1611  4208          DSBINT  UCB$B_DIPL(R5)
                      007B    30  1618  4209          BSBW    START_RECEIVE_BISYNC
                                  161B  4210          ENBINT
                              05  161E  4211          RSB
                                  161F  4212
                                  161F  4213
```

```
                                        161F  4215          .SBTTL  FILLFREELIST - FILL MESSAGE FREE LIST
                                        161F  4216 ;++
                                        161F  4217 ; FILLFREELIST - Fill message block free list
                                        161F  4218 ; ADDFREELIST - Add a buffer to receive list
                                        161F  4219 ;
                                        161F  4220 ; Functional description:
                                        161F  4221 ;
                                        161F  4222 ; This routine fills the receive buffer free list up to the quota specified
                                        161F  4223 ; at device startup.
                                        161F  4224 ;
                                        161F  4225 ; Inputs:
                                        161F  4226 ;
                                        161F  4227 ;       R2 = Buffer address (ADDFREELIST only)
                                        161F  4228 ;       R5 = UCB address
                                        161F  4229 ;
                                        161F  4230 ;       IPL = FIPL
                                        161F  4231 ;
                                        161F  4232 ; Outputs:
                                        161F  4233 ;
                                        161F  4234 ;       R5 = UCB address
                                        161F  4235 ;       R3 is preserved
                                        161F  4236 ;       R1,R2,R4 destroyed.
                                        161F  4237 ;--
                                        161F  4238 FILLFREELIST:                                        ; Fill free list
                                  52    D4  161F  4239          CLRL    R2                              ; Clear buffer address
                                  0B    E0  1621  4240          BBS     #XM$V_STS_ACTIVE,-              ; Continue if device active
                               01 44 A5     1623  4241                  UCB$L_DEVDEPEND(R5),ADDFREELIST ;
                                  05        1626  4242          RSB                                     ;
                                            1627  4243 ADDFREELIST:                                    ; Add to receive buffer list
                                  53    DD  1627  4244          PUSHL   R3                              ; Save registers
                                    42  A5  B1  1629  4245 5$:       CMPW    UCB$W_DEVBUFSIZ(R5),-           ; Can new block be allocated ?
                                 010C   C5      162C  4246                  UCB$W_XG_QUOTA(R5)              ;
                                        37  1A  162F  4247          BGTRU   20$                             ; Br if no - list filled
                                        51  D4  1631  4248          CLRL    R1                              ; Zero size
                                  004C  8F  A1  1633  4249          ADDW3   #RCV_T_DATA+CXB$C_TRAILER,-     ; Compute needed block size
                                51    42  A5      1637  4250                  UCB$W_DEVBUFSIZ(R5),R1          ;
                                        52  D5  163A  4251          TSTL    R2                              ; Buffer allocated already?
                                        09  12  163C  4252          BNEQ    7$                              ; Br if yes
                                00000000'GF  16  163E  4253          JSB     G^EXE$ALONONPAGED               ; Allocate nonpaged memory
                                      1A  50  E9  1644  4254          BLBC    R0,10$                          ; Br if failure
                                08 A2     51  B0  1647  4255 7$:       MOVW    R1,RCV_W_BLKSIZE(R2)            ; Insert block size
                                0A A2     17  90  164B  4256          MOVB    S^#DYN$C_NET,RCV_B_BLKTYPE(R2)  ; Insert block type
                                   0E     A2  B4  164F  4257          CLRW    RCV_W_ERROR(R2)                 ; Clear error status
                               00E4 C5     62  0E  1652  4258          INSQUE  (R2),UCB$Q_XG_FREE(R5)          ; Insert block on list
                                  42    A5  A2  1657  4259          SUBW    UCB$W_DEVBUFSIZ(R5),-           ; Decrement quota
                                 010C   C5      165A  4260                  UCB$W_XG_QUOTA(R5)              ;
                                        52  D4  165D  4261          CLRL    R2                              ; Clear buffer pointer
                                        C8  11  165F  4262          BRB     5$                              ;
                                            1661  4263
                                            1661  4264 10$:      SETBIT  #XM$V_STS_BUFFAIL,-             ; Set buffer alloc failure
                                            1661  4265                  UCB$L_DEVDEPEND(R5)             ;
                                        10  11  1666  4266          BRB     30$                             ;
                                            1668  4267
                                            1668  4268 20$:      CLRBIT  #XM$V_STS_BUFFAIL,-             ; Clear buffer alloc failure
                                            1668  4269                  UCB$L_DEVDEPEND(R5)             ;
                                50  52  D0  166D  4270          MOVL    R2,R0                           ; Set address of buffer
                                        06  13  1670  4271          BEQL    30$                             ; Br if none
```

```
                00000000'GF   16  1672  4272          JSB     G^COM$DRVDEALMEM                        ; Deallocate it
          0167 C5   01   91  1678  4273 30$:     CMPB    #XG$C_PROTYPE_BISYNC,UCB$B_XG_PROTYPE(R5) ; If EQL do not
                    10   13  167D  4274          BEQL    50$                                     ; start receives
                                167F  4275          DSBINT  UCB$B_DIPL(R5)                  ; Disable interrupts
                    7A   10  1686  4276          BSBB    START_RECEIVE                   ; Start then next  receive
                                1688  4277          ENBINT                                  ; Enable interrupts
                    53 8ED0  168B  4278 40$:     POPL    R3                              ; Restore registers
                       05  168E  4279          RSB
                50  01   3C  168F  4280 50$:     MOVZWL  S^#SS$_NORMAL,R0                ; Set for normal return
                    53 8ED0  1692  4281          POPL    R3                              ; Restore registers
                       05  1695  4282          RSB
                                1696  4283
```

```
                        1696  4285          .SBTTL  START_RECEIVE_BISYNC - Start receives in BISYNC mode
                        1696  4286 ;++
                        1696  4287 ; START_RECEIVE_BISYNC - Start receives in BISYNC mode
                        1696  4288 ;
                        1696  4289 ; This routine starts up the receiver in BISYNC mode. The receiver  in
                        1696  4290 ; BISYNC mode works differently from the other modes because in this
                        1696  4291 ; mode there is no way for the device to tell when the receive is
                        1696  4292 ; complete thus we must check each character comming off the line to
                        1696  4293 ; see if it is part of the message we are receiving. In order to do this
                        1696  4294 ; the following has been implemented. At device startup a buffer is
                        1696  4295 ; taken from the freelist and stored in BISYNC$A_RCV_BUFFER, this
                        1696  4296 ; buffer purpose is to act as a holder for characters comming off the wire.
                        1696  4297 ; BISYNC$W_RCV_INDEX will point to the next character from the line
                        1696  4298 ; with in that buffer. As we detect that a new character has come into
                        1696  4299 ; the buffer we will pass that character to the FRAMING ROUTINE provided
                        1696  4300 ; by the starter. The character will either be buffered or not in the
                        1696  4301 ; actual receive buffer (See BISYNC_TIMER for an explanation).
                        1696  4302 ;
                        1696  4303 ;       INPUTS
                        1696  4304 ;               R4 = Address of protocol buffer
                        1696  4305 ;               R5 = UCB address
                        1696  4306 ;               IPL = device IPL
                        1696  4307 ;
                        1696  4308 ;       OUTPUTS
                        1696  4309 ;               R5 = UCB address
                        1696  4310 ;               The receive is started
                        1696  4311 ;
                        1696  4312 ;--
                        1696  4313 START_RECEIVE_BISYNC:
          5A 68 A5   01 E0  1696  4314          BBS     #XG_DS_V_RCVING,UCB$W_DEVSTS(R5),10$     ; If BS not rcving
          55 68 A5   06 E0  169B  4315          BBS     #XG_DS_V_RCV_DONEP,UCB$W_DEVSTS(R5),10$ ; If set then rcv is loaded
             51   24 A5 D0  16A0  4316          MOVL    UCB$L_CRB(R5),R1                        ; Get the CRB address
             54   2C B1 D0  16A4  4317          MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R4            ; Get the CSR address
                  51   06 3C  16A8  4318          MOVZWL  #XG$C_PRI_RCV,R1                        ; Set ind reg primary buffer
                  53   02 9A  16AB  4319          MOVZBL  #XG$C_PRIRCV,R3                         ; Set mapping vector to use
           52   00E4 D5 0F  16AE  4320          REMQUE  @UCB$Q_XG_FREE(R5),R2                   ; See if any RCV buffers free
                     40 1D  16B3  4321          BVS     10$
           00EC C5   52 D0  16B5  4322          MOVL    R2,UCB$L_XG_RCV_INPR(R5)                ; Set rcv buffer
                00F0 C5 B4  16BA  4323          CLRW    UCB$W_XG_RCV_INPR_INDX(R5)              ; Clear index
           50   00D0 C5 D0  16BE  4324          MOVL    UCB$A_XG_PRO_BUFFER(R5),R0              ; Get the protocol buffer
             52   1C A0 D0  16C3  4325          MOVL    BISYNC$A_RCV_BUFFER(R0),R2              ; Give board the buffer to rcv into
                     00B7 30  16C7  4326          BSBW    LOAD_RCV_MPR                            ; Load the receive buffer
        00F2 C5   12 A2 B0  16CA  4327          MOVW    RCV_[_BACC+2(R2),UCB$W_XG_RCV_INIT(R5) ; Save char count of buffer
00F2 C5   02   0F   00 F0  16D0  4328          INSV    #0,#15,#2,UCB$W_XG_RCV_INIT(R5) ; Clear high order bits
                64   04 AA  16D7  4329          BICW2   #XG$M_PRM_SEC,XG$C_RCV_CSR(R4)  ; Make sure we are usin the primayr
                     16DA  4330          SETBIT  #XG_DS_V_RCV_DONEP,UCB$W_DEVSTS(R5) ; Set receiver is loaded
                     16DF  4331          CLRBIT  #XG_DS_V_RCVENB,UCB$W_DEVSTS(R5) ; clear enable receive bit
                     16E4  4332          DSBINT                                  ; Lock out interrupts
          0A 64 A5   05 E0  16EA  4333          BBS     #UCB$V_POWER,UCB$W_STS(R5),20$  ; If BS then powerfail
                64   01 A8  16EF  4334          BISW    #XG$M_ENABLE,XG$C_RCV_CSR(R4)   ; Set the enable bit
                     16F2  4335          ENBINT
                50   01 3C  16F5  4336 10$:     MOVZWL  S^#SS$_NORMAL,R0
                     05 16F8  4337          RSB
                     16F9  4338
                     16F9  4339 20$:     ENBINT
           50   0364 8F 3C  16FC  4340          MOVZWL  #SS$_POWERFAIL,R0
                     05 1701  4341          RSB
```

1702 4342

```
                     1702  4344          .SBTTL  START_RECEIVE - Start any receives
                     1702  4345 ;++
                     1702  4346 ; START_RECEIVE - Start receives
                     1702  4347 ;
                     1702  4348 ; FUNCTIONAL DESCRIPTION:
                     1702  4349 ;
                     1702  4350 ; This routine oversees giving receive buffers to the board. It's job is to
                     1702  4351 ; check and set the copy of the DMF32 receive bits that the driver keeps to
                     1702  4352 ; tell which buffer (primary or secondary) it must load next. If there is no
                     1702  4353 ; receive buffer to give the board, the routine must be smart enough not
                     1702  4354 ; to try to enable the receiver. Unsolicited interrupts can happen if the
                     1702  4355 ; board is enabled and the buffers haven't been loaded. Otherwise the
                     1702  4356 ; routine enables the board after each buffer is loaded whether the board
                     1702  4357 ; is already enabled or not. Remember that this routine may load more
                     1702  4358 ; than one buffer a call.
                     1702  4359 ;
                     1702  4360 ; INPUTS:
                     1702  4361 ;
                     1702  4362 ;       R5 = UCB address
                     1702  4363 ;       IPL = Device IPL
                     1702  4364 ;
                     1702  4365 ; OUTPUTS:
                     1702  4366 ;
                     1702  4367 ;       R5 = UCB address
                     1702  4368 ;
                     1702  4369 ;       R0 - R4 destroyed
                     1702  4370 ;--
                     1702  4371 START_RECEIVE:                                  ; Start receive operation
          01   E0    1702  4372          BBS     #XG_DS_V_RCVING,-             ; If BS then device not
 15 68 A5            1704  4373                  UCB$W_DEVSTS(R5),12$           ; not receiving
51    24 A5   D0     1707  4374          MOVL    UCB$L_CRB(R5),R1              ; Get the CRB address
54    2C B1   D0     170B  4375          MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R4  ; Get the CSR
          05   E0    170F  4376          BBS     #XG_DS_V_LOADRPS,-            ; If BS then load sec buff
  26 68 A5            1711  4377                  UCB$W_DEVSTS(R5),20$          ; else load this buffer
          06   E1    1714  4378 10$:     BBC     #XG_DS_V_RCV_DONEP,-          ; If BC then not loaded
  06 68 A5            1716  4379                  UCB$W_DEVSTS(R5),13$
                      1719  4380
              0042    31    1719  4381          BRW     30$                   ; Jump enable the device
              0055    31    171C  4382 12$:     BRW     35$                   ; Jump to return
                      171F  4383
           51   06    3C    171F  4384 13$:     MOVZWL  #XG$C_PRI_RCV,R1      ; Set ind reg pri buffer
           53   02    9A    1722  4385          MOVZBL  #XG$C_PRIRCV,R3       ; Set mapping vector to use
     52    00E4 D5    0F    1725  4386          REMQUE  @UCB$Q_XG_FREE(R5),R2 ; Get next buffer to input into
                32    1D    172A  4387          BVS     30$                   ; Branch if none
              0052    30    172C  4388          BSBW    LOAD_RCV_MPR          ; Load the receive buffer
     00F0 D5    62    0E    172F  4389          INSQUE  (R2),@UCB$Q_XG_RCV_INPR+4(R5) ; Insert it on the inpr queue
                      AB    1734  4390          BISW2   #<XG_DS_M_LOADRPS!-   ; Set to load sec buff and
                            1735  4391                  XG_DS_M_RCV_DONEP>,-  ; to indicate the pri buff
 68 A5    0060 8F          1735  4392                  UCB$W_DEVSTS(R5)      ; was loaded
                07    E0    173A  4393 20$:     BBS     #XG_DS_V_RCV_DONES,-  ; If BS then sec loaded
        1F 68 A5            173C  4394                  UCB$W_DEVSTS(R5),30$
           51   08    3C    173F  4395          MOVZWL  #XG$C_SEC_RCV,R1      ; Set ind reg sec buffer
           53   03    9A    1742  4396          MOVZBL  #XG$C_SECRCV,R3       ; Set mapping vector to use
     52    00E4 D5    0F    1745  4397          REMQUE  @UCB$Q_XG_FREE(R5),R2 ; Get next buffer to input into
                12    1D    174A  4398          BVS     30$                   ; Branch if none
              0032    30    174C  4399          BSBW    LOAD_RCV_MPR          ; Load the receive buffer
     00F0 D5    62    0E    174F  4400          INSQUE  (R2),@UCB$Q_XG_RCV_INPR+4(R5) ; Insert it on the inpr queue
```

```
                                                D 10
                    20   AA  1754  4401          BICW    #XG_DS_M_LOADRPS,-                  ; Clear to load pri next
                 68 A5       1756  4402                  UCB$W_DEVSTS(R5)
               0080 8F   A8  1758  4403          BISW    #XG_DS_M_RCV_DONES,-                ; Set to indicate sec buff
                 68 A5       175C  4404                  UCB$W_DEVSTS(R5)                    ; was loaded
                    0C   E5  175E  4405  30$:     BBCC    #XG_DS_V_RCVENB,-                   ; If BC then nothing to
              11 68 A5       1760  4406                  UCB$W_DEVSTS(R5),35$                ; give the board
                             1763  4407          DSBINT                                      ; Disable all interrupts
          0A 64 A5    05   E0  1769  4408          BBS     #UCB$V_POWER,UCB$W_STS(R5),40$      ; Branch BS power fail
                64    01   A8  176E  4409          BISW    #XG$M_ENABLE,XG$C_RCV_CSR(R4)       ; Enble XMTer
                             1771  4410          ENBINT                                      ; Restore IPL
                50    01   3C  1774  4411  35$:     MOVZWL  S^#SS$_NORMAL,R0                    ; Set normal return
                             05  1777  4412          RSB
                             1778  4413
                             1778  4414  40$:     ENBINT                                      ; Enable interrupts
                50   0364 8F   3C  177B  4415  45$:     MOVZWL  #SS$_POWERFAIL,R0           ; Set powerfail status
                             05  1780  4416          RSB
                             1781  4417
```

```
                                1781  4419
                                1781  4420 ;++
                                1781  4421 ;LOAD_RCV_MPR - Load receive map registers and character count
                                1781  4422 ;
                                1781  4423 ; This routine gets a free recevie out of the queue and sets up the mapping
                                1781  4424 ; registers and BACC to give to COMBO. In this case the double buffering
                                1781  4425 ; is used so that the board will always have a buffer in which in can receive.
                                1781  4426 ; If too few buffers are allocated or if they are not completed quick enough
                                1781  4427 ; i.e. too few IRP's given to complete the buffer, then no action is taken and
                                1781  4428 ; it is left to the caller to decide what must be done.
                                1781  4429 ;
                                1781  4430 ;       INPUTS:
                                1781  4431 ;               R1 = Address of the indirect register to load
                                1781  4432 ;               R2 = Address of a buffer to receive into
                                1781  4433 ;               R3 = Vector slot to use
                                1781  4434 ;               R4 = CSR address
                                1781  4435 ;               IPL = DIPL
                                1781  4436 ;
                                1781  4437 ;       OUTPUTS
                                1781  4438 ;               R2,R4, and R5 are preserved
                                1781  4439 ;--
                                1781  4440 LOAD_RCV_MPR:
                                1781  4441 ;
                                1781  4442 ; Mark slot in use and create buffer address / character count image
                                1781  4443 ;
                       12   BB  1781  4444         PUSHR   #^M<R1,R4>                      ; Save registers
          54   0110 C543   DE  1783  4445         MOVAL   UCB$Z_XG_VECTOR(R5)[R3],R4      ; Get slot address
        42 A5   0165 C5   A1  1789  4446         ADDW3   UCB$W_XG_MFDLEN(R5),UCB$W_DEVBUFSIZ(R5),-
                   12 A2      178F  4447                 RCV_L_BACC+2(R2)                ; Insert character count
              51   42 A2   DE  1791  4448         MOVAL   RCV_Z_HEADER(R2),R1             ; Get normal buffer address
             10 A2    51   B0  1795  4449         MOVW    R1,RCV_L_BACC(R2)               ; Insert buffer offset
 10 A2   07   09    64   F0  1799  4450         INSV    (R4),#9,#7,RCV_L_BACC(R2)       ; Insert map register data
    53   64   02    07   EF  179F  4451         EXTZV   #7,#2,(R4),R3                   ; Get two high bits
 10 A2   02   1E    53   F0  17A4  4452         INSV    R3,#30,#2,RCV_L_BACC(R2)        ; Insert in BA/CC
                            17AA  4453
                            17AA  4454 ;
                            17AA  4455 ; Build UBA map register data and load map
                            17AA  4456 ;
 7C A5   51    FE00 8F    AB  17AA  4457         BICW3   #^C<VA$M_BYTE>,R1,UCB$W_BOFF(R5); Set byte offset
       7E A5    42 A5    B0  17B1  4458         MOVW    UCB$W_DEVBUFSIZ(R5),UCB$W_BCNT(R5) ; Set byte count
    51   51    15   09   EF  17B6  4459         EXTZV   #VA$V_VPN,#VA$S_VPN,R1,R1       ; Get virtual page number
     50   00000000'GF   D0  17BB  4460         MOVL    G^MMG$GL_SPTBASE,R0             ; Get SPT address
          78 A5   6041   DE  17C2  4461         MOVAL   (R0)[R1],UCB$L_SVAPTE(R5)       ; Set address of SPT entry
            51   24 A5   D0  17C7  4462         MOVL    UCB$L_CRB(R5),R1                ; Get CRB address
            34 A1   64   D0  17CB  4463         MOVL    (R4),CRB$L_INTD+VEC$W_MAPREG(R1); Set starting map reg number
                    52   DD  17CF  4464         PUSHL   R2                              ; Save registers
          00000000'GF   16  17D1  4465         JSB     G^IOC$LOADUBAMAPA               ; Load the map registers
                    52 8ED0  17D7  4466         POPL    R2                              ; Restore register
                    12   BA  17DA  4467         POPR    #^M<R1,R4>                      ; Restore registers
            04 A4   51   B0  17DC  4468         MOVW    R1,XG$C_MISC_REG(R4)            ; Load into indirect register
                 10 A2   B0  17E0  4469         MOVW    RCV_L_BACC(R2),-                ; Load buffer address
                 06 A4       17E3  4470                 XG$C_IND_ADDR(R4)
                 12 A2   B0  17E5  4471         MOVW    RCV_L_BACC+2(R2),-              ; and character count
                 06 A4       17E8  4472                 XG$C_IND_ADDR(R4)
                1000 8F   A8  17EA  4473         BISW    #XG_DS_M_RCVENB,-               ; Set to indicate that there
                   68 A5      17EE  4474                 UCB$W_DEVSTS(R5)                ; is work for the board
                   50   01   3C  17F0  4475         MOVZWL  S^#SS$_NORMAL,R0                ; Set normal return
```

```
05  17F3  4476 10$:     RSB
    17F4  4477
```

```
                        17F4  4479                .SBTTL  INTERRUPTS
                        17F4  4480 ;++
                        17F4  4481 ; TRANSMIT_INTR
                        17F4  4482 ;
                        17F4  4483 ; FUNCTIONAL DESCRIPTION:
                        17F4  4484 ;
                        17F4  4485 ; This routine is called when an interrupt from the transmit vector occurs.
                        17F4  4486 ; The action is to check for errors on the tranmsit. If they occur then the
                        17F4  4487 ; device must be shut down because only fatal errors occur on transmits.
                        17F4  4488 ; Otherwise it makes sure the board and the driver agree as to what bits
                        17F4  4489 ; should be set and if all is OK then it puts the buffer on the POST queue
                        17F4  4490 ; to be handled at fork processing time. Finally, it tries to start up the
                        17F4  4491 ; next transmit. When the line is running Half duplex or in trib mode
                        17F4  4492 ; ddcmp only then if this is the last message to be transmitted in this
                        17F4  4493 ; selection interval the XMTer is shut off. For half duplex this includes
                        17F4  4494 ; starting up a TQE on waiting for CTS to be dropped because on real
                        17F4  4495 ; modems this may not happen right away. The modem must have dropped CTS
                        17F4  4496 ; before we reasscert RTS otherwise it gets all confused.
                        17F4  4497 ;
                        17F4  4498 ;       INPUTS  00(SP) = IDB address
                        17F4  4499 ;               04(SP) = R0 - R5
                        17F4  4500 ;
                        17F4  4501 ;       OUTPUTS
                        17F4  4502 ;               An entry is added to the post queue.
                        17F4  4503 ;
                        17F4  4504 ;--
                        17F4  4505 TRANSMIT_INTR:                                       ; Transmit interrupt
                        17F4  4506         .ENABL  LSB
    54    9E    D0      17F4  4507         MOVL    @(SP)+,R4                            ; Get IDB address
55    18    A4    D0      17F7  4508         MOVL    IDB$L_UCBLST(R4),R5                  ; Get Ucb address
            0B    E0      17FB  4509         BBS     #XM$V_STS_ACTIVE,-                   ; Branch BS device active
      03    44    A5      17FD  4510                 UCB$L_DEVDEPEND(R5),5$
          01F8    31      1800  4511         BRW     INTEXIT                              ; If device not active
      0104    C5    D4    1803  4512 5$:     CLRL    UCB$L_XG_XMTEND(R5)                  ; Reset timeout
      54    64    D0      1807  4513         MOVL    (R4),R4                              ; Get CSR address
        02    A4    B0      180A  4514         MOVW    XG$C_XMT_CSR(R4),-                   ; Save last XMT csr
      0124    C5            180D  4515                 UCB$W_XG_XMTCSR(R5)
04    A4    04    B0      1810  4516         MOVW    #XG$C_MODEM,XG$C_MISC_REG(R4)        ; Load the IR with dsc reg
        06    A4    B0      1814  4517         MOVW    XG$C_IND_ADDR(R4),-                  ; Save last DSC csr
      012A    C5            1817  4518                 UCB$W_XG_DSC(R5)
            0E    E1      181A  4519         BBC     #XG$V_ERROR,-                        ; Branch BC no fatal error
1A    0124    C5            181C  4520                 UCB$W_XG_XMTCSR(R5),8$               ; on the transmit
04    A4    02    B0      1820  4521         MOVW    #XG$C_XMT_ERR,XG$C_MISC_REG(R4)      ; Load the IR with error reg
        06    A4    9B      1824  4522         MOVZBW  XG$C_IND_ADDR(R4),-                  ; Get the error only low byte
      0128    C5            1827  4523                 UCB$W_XG_XMTERR(R5)                  ; is significant
            12    13      182A  4524         BEQL    10$                                  ; if no error set (XMT's were aborte
            00    E0      182C  4525         BBS     #XG$V_MSG_LEN,-                      ; If message length error then
0C    0128    C5            182E  4526                 UCB$W_XG_XMTERR(R5),10$              ; non-fatal
                        1832  4527
                        1832  4528         .IF     DF ERR$$$
                        1832  4529         bbc     #xg$v_latency_xmt,-                  ; if BC fatal error not due
                        1832  4530                 ucb$w_xg_xmterr(R5),6$               ; to a latency problem
                        1832  4531         incw    ucb$w_xg_xmtlat(R5)                  ; inc the latency error count
                        1832  4532         brb     7$
                        1832  4533 6$:     incw    ucb$w_xg_xmtnxm(r5)                  ; inc nxm count
                        1832  4534         .ENDC   ;DF ERR$$$
                        1832  4535
```

```
                54   D4  1832  4536 7$:     CLRL    R4                                  ; Set fatal error on device
              01D3   30  1834  4537          BSBW    SCHED_FORK                          ; Schedule a fork
              01C1   31  1837  4538          BRW     INTEXIT
                     183A  4539
            0128 C5  B4  183A  4540 8$:     CLRW    UCB$W_XG_XMTERR(R5)                 ; Set no error
                 0F  E1  183E  4541 10$:    BBC     #XG$V_DONE_P,-                      ; If BC then may be from
         59 0124 C5          1840  4542              UCB$W_XG_XMTCSR(R5),30$             ; Data.Set.Change
                 53  D4  1844  4543          CLRL    R3                                  ; Set to set up prim BACC
      52   00FC C543  D0  1846  4544          MOVL    UCB$Z_XG_XMT_INPR(R5)[R3],R2        ; Get the buffer to complete
                 1E  13  184C  4545          BEQL    15$                                 ; If EQL nothing ot complete
            0164 C5  97  184E  4546          DECB    UCB$B_XG_XMTCNT(R5)                 ; Decr the no of outstndng XMTs
          00FC C543  D4  1852  4547          CLRL    UCB$Z_XG_XMT_INPR(R5)[R3]           ; Clear the address from slot
                     1857  4548          CLRBIT  #XG_DS_V_XMT_DONEP,UCB$W_DEVSTS(R5) ; Clear indicating that the
                     185C  4549                                                      ; XMT CSR is now free to use
                 02  E0  185C  4550          BBS     #XMTQ$V_CONTROL,-                   ; If BS then control msg do
          0B 1F A2          185E  4551              XMTQ$B_FLAG(R2),15$                 ; put on post queue
            0128 C5  B0  1861  4552          MOVW    UCB$W_XG_XMTERR(R5),-               ; Set error status
                1D A2          1865  4553              XMTQ$Q_ERROR(R2)
      00F8 D5    62  0E  1867  4554          INSQUE  (R2),@UCB$Q_XG_POST+4(R5)           ; Insert it onto the post Q
                 00  E0  186C  4555 15$:    BBS     #XG_DS_V_XMTING,-                   ; If BS then time to turn link
             2F 68 A5          186E  4556              UCB$W_DEVSTS(R5),40$                ; and drop RTS
              52   0A  9A  1871  4557 18$:    MOVZBL  #XG$C_PRI_XMT,R2                    ; Set ind reg addr of prim
                 F4C1  30  1874  4558          BSBW    LOAD_XMT_MPR                        ; Load map registers and BACC
                17 50  E9  1877  4559          BLBC    R0,20$                              ; No tranmsit given to board
                     187A  4560          SETBIT  #XG_DS_V_XMT_DONEP,UCB$W_DEVSTS(R5) ; Set indicating that the
                     187F  4561                                                      ; XMT CSR is not free to use
                     187F  4562          DSBINT                                      ; Sync to highest IPL
         10 64 A5    05  E0  1885  4563          BBS     #UCB$V_POWER,UCB$W_STS(R5),25$      ; If BS then power fail occured
            02 A4    01  A8  188A  4564          BISW    #XG$M_ENABLE,XG$C_XMT_CSR(R4)       ; Else enable transmitter
                     188E  4565          ENBINT                                      ; Enable interrupts
              54   01  3C  1891  4566 20$:    MOVZWL  S^#SS$_NORMAL,R4                    ; Set no errors
                 0173  30  1894  4567          BSBW    SCHED_FORK
                 0161  31  1897  4568          BRW     INTEXIT
                     189A  4569
                     189A  4570 25$:    ENBINT                                      ; Enable interrupts
                 0165  31  189D  4571 30$:    BRW     INTERR
                     18A0  4572
               0120 C5  94  18A0  4573 40$:    CLRB    UCB$B_XG_XSTATE(R5)                 ; Assume IDLE state
            04 A4    04  B0  18A4  4574          MOVW    #XG$C_MODEM,XG$C_MISC_REG(R4)       ; Set to get modem indirect reg
      06 A4    1000 8F  AA  18A8  4575          BICW    #XG$M_RTS,XG$C_IND_ADDR(R4)         ; Clear RTS
        54 44 A5    07  E0  18AE  4576          BBS     #XM$V_CHR_TRIB,UCB$L_DEVDEPEND(R5),50$ ; If BS then ignore chck
            04 A4    04  B0  18B3  4577          MOVW    #XG$C_MODEM,XG$C_MISC_REG(R4)       ; Set to get modem indirect reg
                     18B7  4578          TIMEWAIT #1,#XG$M_CTS,XG$C_IND_ADDR(R4),W,EQL
                  29 50  E8  18DB  4579          BLBS    R0,50$                              ; If LBS then CTS was dropped
             0120 C5  06  90  18DE  4580          MOVB    #XG$C_DRPCTS,UCB$B_XG_XSTATE(R5)    ; Set wait for CTS to drop
           28 68 A5  0E  E2  18E3  4581          BBSS    #XG_DS_V_CTSTQE_RUN,UCB$W_DEVSTS(R5),60$ ; If BS timer is q'd
                 55  DD  18E8  4582          PUSHL   R5                                  ; Save R5 across fork
            55   0094 C5  DE  18EA  4583          MOVAL   UCB$L_XG_TQE(R5),R5                 ; Get the CTS TQE block
             0B A5   06  90  18EF  4584          MOVB    #IPL$_QUEUEAST,TQE$B_RQTYPE(R5)     ; Set IPL of fork process
                00'AF  9F  18F3  4585          PUSHAB  B^DRPCTSRET                         ; Set return address
               F5CF  CF  3F  18F6  4586          PUSHAW  W^START_TIMER                       ; Set addr of CTS TQE start
          00000000'GF  17  18FA  4587          JMP     G^EXE$FORK
                     1900  4588
                     1900  4589 DRPCTSRET:
             0B A5   01  90  1900  4590          MOVB    #TQE$C_SSSNGL,TQE$B_RQTYPE(R5)      ; Set single shot timer
                 55 8ED0  1904  4591          POPL    R5                                  ; Restore R5
                54   01  3C  1907  4592 50$:    MOVZWL  S^#SS$_NORMAL,R4
```

```
00FD   30  190A  4593          BSBW    SCHED_FORK
00EB   31  190D  4594          BRW     INTEXIT
           1910  4595
           1910  4596 60$:     BUG_CHECK       NOBUFPCKT,FATAL
           1914  4597          .DSABL  LSB
           1914  4598
```

```
                          1914  4600
                          1914  4601 ;++
                          1914  4602 ;RECEIVE_INTR
                          1914  4603 ;
                          1914  4604 ; FUNCTINAL DESCRIPTION:
                          1914  4605 ;
                          1914  4606 ; This routine is called when an interrupt from the receive vector occurs. It
                          1914  4607 ; checks for errors and if the error is non-fatal to the device then the  error
                          1914  4608 ; is squirled away in the buffer to be handled at fork processing time. It does
                          1914  4609 ; checking on the driver receive bits against the DMF receive bits to be sure
                          1914  4610 ; that all is still kosher. Next it removes a buffer from the receive
                          1914  4611 ; inprogress queue checks to be sure that it is the buffer expected and if OK
                          1914  4612 ; gives the buffer to the post queue to be handled at fork processing time.
                          1914  4613 ; Finally, it gives a receive buffer to the device.
                          1914  4614 ;
                          1914  4615 ;       INPUTS  00(SP) = IDB address
                          1914  4616 ;               04(SP) = R0 - R5
                          1914  4617 ;
                          1914  4618 ;       OUTPUTS
                          1914  4619 ;               An entry is added to the post queue.
                          1914  4620 ;
                          1914  4621 ;
                          1914  4622 RECEIVE_INTR:
    54    9E    D0        1914  4623          MOVL    @(SP)+,R4                       ; Get IDB address
55    18 A4    D0        1917  4624          MOVL    IDB$L_UCBLST(R4),R5             ; Get Ucb address
       0B    E0           191B  4625          BBS     #XM$V_STS_ACTIVE,-              ; Branch BC device active
  03 44 A5                191D  4626                  UCB$L_DEVDEPEND(R5),5$
      00D8    31           1920  4627 2$:      BRW     INTEXIT                         ; Branch device not active
    52    02    9A        1923  4628 5$:      MOVZBL  #2,R2                           ; Set for max # of tms to loop
    54    64    D0        1926  4629          MOVL    (R4),R4                         ; Get CSR address
          64    B0        1929  4630          MOVW    XG$C_RCV_CSR(R4),-              ; Save the receive csr
        0122 C5           192B  4631                  UCB$W_XG_RCVCSR(R5)
                          192E  4632 ;
                          192E  4633 ; If the device is running in BISYNC mode then we should never get a receive
                          192E  4634 ; interrupt. If we do get one then just dismiss it.
                          192E  4635 ;
  0167 C5    01    91      192E  4636 10$:     CMPB    #XG$C_PROTYPE_BISYNC,UCB$B_XG_PROTYPE(R5)       ; BISYNC?
             EB    13      1933  4637          BEQL    2$                              ; If EQL then dismiss intr
  64    4000 8F    B3      1935  4638          BITW    #XG$M_ERROR,XG$C_RCV_CSR(R4)    ; Branch NEQ no fatal error
             21    13      193A  4639          BEQL    13$                             ; on the receive
             01    B0      193C  4640          MOVW    #XG$C_RCV_ERR,-                 ; Load the addr of error IR
           04 A4           193E  4641                  XG$C_MISC_REG(R4)
           06 A4    B0     1940  4642          MOVW    XG$C_IND_ADDR(R4),-             ; Get the error
        0126 C5           1943  4643                  UCB$W_XG_RCVERR(R5)
        1000 8F    A8     1946  4644          BISW    #XG_DS_M_RCVENB,-               ; Set to restart rcv'r won't
          68 A5           194A  4645                  UCB$W_DEVSTS(R5)                ; get retsrted in fatal error
                          194C  4646                                                  ; device will be shut down
                    B3    194C  4647          BITW    #<XG$M_BCC_ERR!XG$M_BUFOVR!-    ; If any set error not fatal
                          194D  4648                  XG$M_RES_BIT_CNT!XG$M_ABORT>,-
0126 C5    0768 8F        194D  4649                  UCB$W_XG_RCVERR(R5)
                0C    12  1953  4650          BNEQ    15$
                          1955  4651
                          1955  4652          .IF     DF_ERR$$
                          1955  4653          bbc     #xg$v_latency_rcv,-             ; if BC fatal error not due
                          1955  4654                  ucb$w_xg_rcverr(r5),110$        ; to a latency problem
                          1955  4655          incw    ucb$w_xg_rcvlat(r5)             ; inc the latency error count
                          1955  4656          brb     120$
```

```
                                                    K 10
                                1955  4657 110$:    incw    ucb$w_xg_rcvnxm(r5)
                                1955  4658 120$:
                                1955  4659          .ENDC   ;DF ERR$$$
                                1955  4660
                          54    D4  1955  4661          CLRL    R4                              ; Set fatal error on device
                        00B0    30  1957  4662          BSBW    SCHED_FORK                      ; Schedule the fork
                        009E    31  195A  4663          BRW     INTEXIT
                                    195D  4664
                      0126 C5   B4  195D  4665 13$:     CLRW    UCB$W_XG_RCVERR(R5)             ; Set no error
                          04    E0  1961  4666 15$:     BBS     #XG_DS_V_RCVPS,-                ; If BS then complete sec buffr
                     3B 68 A5       1963  4667                  UCB$W_DEVSTS(R5),20$
                64    8000 8F   B3  1966  4668          BITW    #XG$M_DONE_P,XG$C_RCV_CSR(R4)   ; If EQL buffer not compl
                          6C    13  196B  4669          BEQL    30$                             ; unexpected condition
                51    00EC D5   0F  196D  4670          REMQUE  @UCB$Q_XG_RCV_INPR(R5),R1       ; Get next buffer to complete
                          28    1D  1972  4671          BVS     17$                             ; If BS then unexpected interrupt
                      0040 8F   AA  1974  4672 16$:     BICW    #XG_DS_M_RCV_DONEP,-            ; Set that the buffer can be
                        68 A5       1978  4673                  UCB$W_DEVSTS(R5)                ; reloaded
                      0126 C5   B0  197A  4674          MOVW    UCB$W_XG_RCVERR(R5),-           ; Set any errors
                        0E A1       197E  4675                  RCV_W_ERROR(R1)
                  04 A4     07    B0  1980  4676          MOVW    #XG$C_PRI_RCV1,XG$C_MISC_REG(R4) ; Set up to get bytes xfered
                50    06 A4     B0  1984  4677          MOVW    XG$C_IND_ADDR(R4),R0            ; Get bytes transfered
                50    C000 8F   AA  1988  4678          BICW    #^X<C000>,R0                    ; Clear high order bits
          0C A1   42 A5     50    A3  198D  4679          SUBW3   R0,UCB$W_DEVBUFSIZ(R5),RCV_W_MSGSIZ(R1) ; Get actual xfered
                00F8 D5     61    0E  1993  4680          INSQUE  (R1),@UCB$Q_XG_POST+4(R5)       ; Insert into post queue
                          10    A8  1998  4681          BISW    #XG_DS_M_RCVPS,-                ; Flip the bit to say that the
                        68 A5       199A  4682                  UCB$W_DEVSTS(R5)                ; next inter is expected on Sec
                        8F 52   F5  199C  4683 17$:     SOBGTR  R2,10$                          ; If GTR more work
                          38    11  199F  4684          BRB     30$                             ; Else return
                64    0080 8F   B3  19A1  4685 20$:     BITW    #XG$M_DONE_S,XG$C_RCV_CSR(R4)   ; If EQL sec buf has not comp'd
                          31    13  19A6  4686          BEQL    30$                             ; Branch unexpected interr
                51    00EC D5   0F  19A8  4687          REMQUE  @UCB$Q_XG_RCV_INPR(R5),R1       ; Get next buffer to complete
                          ED    1D  19AD  4688          BVS     17$                             ; If VS then unexpected interr
                      0080 8F   AA  19AF  4689          BICW    #XG_DS_M_RCV_DONES,-            ; Set that the sec buff can be
                        68 A5       19B3  4690                  UCB$W_DEVSTS(R5)                ; reloaded
                      0126 C5   B0  19B5  4691          MOVW    UCB$W_XG_RCVERR(R5),-           ; Set any errors
                        0E A1       19B9  4692                  RCV_W_ERROR(R1)
                  04 A4     09    B0  19BB  4693          MOVW    #XG$C_SEC_RCV1,XG$C_MISC_REG(R4) ; Set up to get bytes xfered
                50    06 A4     B0  19BF  4694          MOVW    XG$C_IND_ADDR(R4),R0            ; Get bytes transfered
                50    C000 8F   AA  19C3  4695          BICW    #^X<C000>,R0                    ; Clear high order bits
          0C A1   42 A5     50    A3  19C8  4696          SUBW3   R0,UCB$W_DEVBUFSIZ(R5),RCV_W_MSGSIZ(R1) ; Get actual xfered
                00F8 D5     61    0E  19CE  4697          INSQUE  (R1),@UCB$Q_XG_POST+4(R5)       ; Insert into post queue
                          10    AA  19D3  4698          BICW    #XG_DS_M_RCVPS,-                ; Flip the bit to say that the
                        68 A5       19D5  4699                  UCB$W_DEVSTS(R5)                ; next inter is expected on pri
                          C3    11  19D7  4700          BRB     17$                             ; Any more work?
                                    19D9  4701
                      FD26    30    19D9  4702 30$:     BSBW    START_RECEIVE                   ; Start any receives
                   26 50      E9    19DC  4703          BLBC    R0,INTERR                       ; If clear then problem
                54    01      3C    19DF  4704          MOVZWL  S^#SS$_NORMAL,R4                ; Set no errors
                              D3    19E2  4705 35$:     BITL    #<XM$M_CHR_HDPLX!-              ; If EQL then mode of device is
                                    19E3  4706                  XM$M_CHR_TRIB>,-                ; not Half dup or Trib
          44 A5   00000084 8F       19E3  4707                  UCB$L_DEVDEPEND(R5)
                          06    13  19EA  4708          BEQL    36$
                      0164 C5   95  19EC  4709          TSTB    UCB$B_XG_XMTCNT(R5)             ; Else check to make sure all
                          02    14  19F0  4710          BGTR    38$                             ; outstandng XMT's have comp'd
                          16    10  19F2  4711 36$:     BSBB    SCHED_FORK                      ; Fork
                          05    11  19F4  4712 38$:     BRB     INTEXIT                         ; Exit interrupt
                                    19F6  4713
```

```
                     19F6  4714 40$:    ENBINT                                  ; Retore IPL
          0A   11    19F9  4715         BRB     INTERR                          ; Exit setting error
                     19FB  4716 ;
                     19FB  4717 ; Exit interrupt
                     19FB  4718 ;
                     19FB  4719 INTEXIT:                                        ; Exit interrupt
     50   8E   7D    19FB  4720         MOVQ    (SP)+,R0
     52   8E   7D    19FE  4721         MOVQ    (SP)+,R2
     54   8E   7D    1A01  4722         MOVQ    (SP)+,R4
                02    1A04  4723         REI
                     1A05  4724 ;
                     1A05  4725 ; An unexpected interrupt occured.  Since there is no NOP function to initiate,
                     1A05  4726 ; the device must be shutdown.
                     1A05  4727 ;
                     1A05  4728 ;
                     1A05  4729 INTERR:                                         ;
   0685   30         1A05  4730         BSBW    TIMEOUT                         ; Fake a timeout error
     F1   11         1A08  4731         BRB     INTEXIT
                     1A0A  4732 ;
                     1A0A  4733 ;       INPUT   R4 = Device status
                     1A0A  4734 ;               R5 = UCB address
                     1A0A  4735 ;
                     1A0A  4736 SCHED_FORK:
          03   E2    1A0A  4737         BBSS    #XG_DS_V_FORK_PEND,-            ; Set fork pending
  09 68 A5           1A0C  4738                 UCB$W_DEVSTS(R5),10$
       20'AF   9F    1A0F  4739         PUSHAB  B^FORK_DONE                     ; Push return address
00000000'GF   17    1A12  4740         JMP     G^EXE$FORK                      ; Fork down to  FIPL
          04 54   E8  1A18  4741 10$:    BLBS    R4,20$                          ; If BS then no need to set err
 14 A5    54   D0    1A1B  4742         MOVL    R4,UCB$L_FR4(R5)                ; Else set error in fork block
                05    1A1F  4743 20$:    RSB
                     1A20  4744
                     1A20  4745
```

```
                                1A20  4747          .SBTTL  FORKDONE - Fork process
                                1A20  4748 ;++
                                1A20  4749 ; FORKDONE - Fork process
                                1A20  4750 ;
                                1A20  4751 ; FUNCTIONAL DESCRIPTION:
                                1A20  4752 ;
                                1A20  4753 ; This routine is entered at device fork level when a buffer done
                                1A20  4754 ; has occured.
                                1A20  4755 ;
                                1A20  4756 ; INPUTS:
                                1A20  4757 ;
                                1A20  4758 ;       R4 = Status from interrupt
                                1A20  4759 ;       R5 = UCB address
                                1A20  4760 ;
                                1A20  4761 ;       IPL = FIPL
                                1A20  4762 ;
                                1A20  4763 ; OUTPUTS:
                                1A20  4764 ;
                                1A20  4765 ;       R5 = UCB address
                                1A20  4766 ;       R6 - R9 are preserved.
                                1A20  4767 ;
                                1A20  4768 ;       If a receive I/O request is pending, the receive is completed.
                                1A20  4769 ;       Otherwise, queue the message for a future I/O.
                                1A20  4770 ;
                                1A20  4771 ;--
                                1A20  4772 FORK_DONE:
                      5D 54  E9 1A20  4773          BLBC    R4,23$                          ; If LBC then fatal error
                                1A23  4774          CLRBIT  #XG_DS_V_FORK_PEND,-            ; Clear fork pending
                                1A23  4775                  UCB$W_DEVSTS(R5)
                                1A28  4776
                50  0167 C5  9A  1A28  4777          MOVZBL  UCB$B_XG_PROTYPE(R5),R0         ; Get protocol type
                                1A2D  4778          CASE    R0,TYPE=B,<-                    ; Case on protocol type
                                1A2D  4779                  FORK_DONE_LAPB,-                ; LAPB
                                1A2D  4780                  40$,-                           ; BISYNC
                                1A2D  4781                  >                               ; Fall thru on DDCMP
                    03C0 8F     BB  1A35  4782          PUSHR   #^M<R6,R7,R8,R9>            ; Save registers
                52  00F4 D5     0F  1A39  4783 5$:      REMQUE  @UCB$Q_XG_POST(R5),R2       ; Get next buffer to complete
                        2A     1D  1A3E  4784          BVS     20$                         ; If VS then no buffer found
                  13    0A A2  91  1A40  4785          CMPB    IRP$B_TYPE(R2),S^#DYN$C_BUFIO ; If NEQ then not XMT buffer
                           08  12  1A44  4786          BNEQ    10$
                         028B  30  1A46  4787          BSBW    TRANSMIT_DONE               ; Complete the transmit
                        43 50  E9  1A49  4788          BLBC    R0,25$                      ; If BC then error on buffer
                           0C  11  1A4C  4789          BRB     15$
                  17    0A A2  91  1A4E  4790 10$:      CMPB    IRP$B_TYPE(R2),S^#DYN$C_NET ; If EQL then RCV buffer
                           4A  12  1A52  4791          BNEQ    30$
                         012D  30  1A54  4792          BSBW    RECEIVE_DONE                ; Complete the Receive
                        35 50  E9  1A57  4793          BLBC    R0,25$                      ; If BC then error on buffer
                      0120 C5  95  1A5A  4794 15$:      TSTB    UCB$B_XG_XSTATE(R5)         ; If NEQ then transmitter going
                           08  12  1A5E  4795          BNEQ    17$
                           51  D4  1A60  4796          CLRL    R1                          ; Set no status for start xmt
                         F244  30  1A62  4797          BSBW    START_TRANSMIT              ; Else try to start up a xmt
                        27 50  E9  1A65  4798          BLBC    R0,25$                      ; Branch BC error
                           CF  11  1A68  4799 17$:      BRB     5$
                      0120 C5  95  1A6A  4800 20$:      TSTB    UCB$B_XG_XSTATE(R5)         ; If NEQ then transmitter going
                           08  12  1A6E  4801          BNEQ    22$
                           51  D4  1A70  4802          CLRL    R1                          ; Set no status for start xmt
                         F234  30  1A72  4803          BSBW    START_TRANSMIT              ; Else try to start up a xmt
```

```
                17 50    E9  1A75  4804              BLBC    R0,25$                          ; Branch BC error
             50    01    3C  1A78  4805  22$:        MOVZWL  S^#SS$_NORMAL,R0                ; Set status
          03C0 8F       BA  1A7B  4806              POPR    #^M<R6,R7,R8,R9>                ; Restore registers
                        05  1A7F  4807              RSB                                     ; Return
                            1A80  4808
          0082 C5       B6  1A80  4809  23$:        INCW    UCB$W_ERRCNT(R5)                ; Adjust count
                            1A84  4810              SETBIT  #XM$V_ERR_FATAL,UCB$L_DEVDEPEND(R5) ; Set fatal error
                05CF    30  1A89  4811              BSBW    POKE_OSER                       ; Tell owner
                066F    31  1A8C  4812              BRW     SHUTDOWN_LINE                   ; Shutdown the line
                            1A8F  4813                                                      ; shut down device
                            1A8F  4814
          03C0 8F       BA  1A8F  4815  25$:        POPR    #^M<R6,R7,R8,R9>                ; Restore registers
                    16  E1  1A93  4816              BBC     #XM$V_ERR_TRIB,-                ; Assume trib shutdown
          03 44 A5          1A95  4817                      UCB$L_DEVDEPEND(R5),28$
                073B    31  1A98  4818              BRW     SHUTDOWN_CIRCUIT
                0660    31  1A9B  4819  28$:        BRW     SHUTDOWN_LINE                   ; Br to shut down the device
                            1A9E  4820
                            1A9E  4821  30$:        BUG_CHECK NOBUFPCKT,FATAL               ; Else fatal, error
                            1AA2  4822
                0094    31  1AA2  4823  40$:        BRW     FORK_DONE_BISYNC
                            1AA5  4824
```

```
                                                B 11
                     1AA5  4826          .SBTTL  FORK_DONE_LAPB
                     1AA5  4827 ;++
                     1AA5  4828 ; FORK_DONE_LAPB - Fork process for LAPB
                     1AA5  4829 ;
                     1AA5  4830 ; This routine does fork processing for LAPB.
                     1AA5  4831 ;
                     1AA5  4832 ;       INPUTS  R5 = UCB address
                     1AA5  4833 ;
                     1AA5  4834 ;       IPL = FIPL
                     1AA5  4835 ;
                     1AA5  4836 ;       OUTPUTS R5-R8 are preserved
                     1AA5  4837 ;
                     1AA5  4838 ;--
                     1AA5  4839 FORK_DONE_LAPB:
 51   00F4 D5   0F   1AA5  4840 5$:     REMQUE  @UCB$Q_XG_POST(R5),R1           ; Get next buffer to complete
           63   1D   1AAA  4841         BVS     40$
   13    0A A1   91   1AAC  4842         CMPB    IRP$B_TYPE(R1),S^#DYN$C_BUFIO   ; If NEQ then not XMT buffer
           26   12   1AB0  4843         BNEQ    20$
   53    0C A1   D0   1AB2  4844         MOVL    XMTQ$L_IRP(R1),R3               ; Get IRP address
   50    1A A1   B0   1AB6  4845         MOVW    XMTQ$W_MSGSIZE(R1),R0           ; Get size of transfer
 50   50    10   78   1ABA  4846         ASHL    #16,R0,R0                       ; Move to high word
           07   12   1ABE  4847         BNEQ    10$                             ; If NEQ then ok
 50   0054 8F   B0   1AC0  4848         MOVW    #SS$_CTRLERR,R0                 ; Set error
           08   11   1AC5  4849         BRB     15$
      50   01   B0   1AC7  4850 10$:    MOVW    S^#SS$_NORMAL,R0                ; Set normal return assume suc
35 68 A5   0F   E4   1ACA  4851         BBSC    #XG_DS_V_CLEAN,UCB$W_DEVSTS(R5),35$ ; If BC then do not abort
   38 A3   50   D0   1ACF  4852 15$:    MOVL    R0,IRP$L_MEDIA(R3)              ; Set status and size
        01C3   30   1AD3  4853         BSBW    TRANSMIT_IO_DONE                ; Complete the I/O
           CD   11   1AD6  4854         BRB     5$
   17    0A A1   91   1AD8  4855 20$:    CMPB    IRP$B_TYPE(R1),S^#DYN$C_NET     ; If NEQ then not RCV buf
           57   12   1ADC  4856         BNEQ    50$
      0708 8F   B3   1ADE  4857         BITW    #<XG$M_BCC_ERR!XG$M_RES_BIT_CNT>,- ; If neq then no error
        0E A1        1AE2  4858                 RCV_W_ERROR(R1)
           2D   12   1AE4  4859         BNEQ    45$
53   00DC D5   0F   1AE6  4860         REMQUE  @UCB$Q_XG_RCVS(R5),R3           ; Get free I/O packet
           10   1D   1AEB  4861         BVS     30$                             ; If VS then none waiting
      52   51   D0   1AED  4862         MOVL    R1,R2                           ; Set up buffer for done
   62    42 A2   DE   1AF0  4863         MOVAL   RCV_Z_HEADER(R2),(R2)           ; Set address of data
   50    0C A2   3C   1AF4  4864         MOVZWL  RCV_W_MSGSIZ(R2),R0             ; Get size of transfer
        0161   30   1AF8  4865         BSBW    FINISH_RCV_IO_LAPB              ; Complete the request
           A8   11   1AFB  4866         BRB     5$
00D8 D5    61   0E   1AFD  4867 30$:    INSQUE  (R1),@UCB$Q_XG_ATTN+4(R5)       ; Queue buffer to await I/O
           A1   11   1B02  4868         BRB     5$                              ; Get next buffer to complete
      50   2C   B0   1B04  4869 35$:    MOVW    #SS$_ABORT,R0                   ; Else set abort status
54   00D0 C5   D0   1B07  4870         MOVL    UCB$A_XG_PRO_BUFFER(R5),R4      ; Set protocol buffer address
        F48B   30   1B0C  4871         BSBW    CLEAN_FORK_ENTRY                ; Clean outstanding on device
      50   01   3C   1B0F  4872 40$:    MOVZWL  S^#SS$_NORMAL,R0                ; Set status
                05   1B12  4873         RSB
                     1B13  4874
                     1B13  4875 ; On error come here
50   00D0 C5   D0   1B13  4876 45$:    MOVL    UCB$A_XG_PRO_BUFFER(R5),R0      ; Get protocol buffer
     16 A0   96   1B18  4877         INCB    LAPB$B_DEI(R0)                  ; Incr error counter
           03   1E   1B1B  4878         BCC     48$                             ; If BC then carry not set
     16 A0   97   1B1D  4879         DECB    LAPB$B_DEI(R0)                  ; Else set to max value
00E8 D5    61   0E   1B20  4880 48$:    INSQUE  (R1),@UCB$Q_XG_FREE+4(R5)       ; Return the buff to free list
                     1B25  4881         DSBINT  UCB$B_DIPL(R5)                  ; Disable interrupts
        FBD3   30   1B2C  4882         BSBW    START_RECEIVE                   ; Start any receives
```

```
                1B2F  4883          ENBINT                                  ; Enable interrupts
FF70    31      1B32  4884          BRW     5$                              ; Complete next req
                1B35  4885
                1B35  4886 50$:     BUG_CHECK NOBUFPCKT,FATAL               ; fatal error
                1B39  4887
```

```
                                1B39  4889          .SBTTL  FORK_DONE_BISYNC - Fork routine for BISYNC mode
                                1B39  4890 ;++
                                1B39  4891 ; FORK_DONE_BISYNC - Fork routine for BISYNC mode
                                1B39  4892 ;
                                1B39  4893 ; This routine does the fork processing for BISYNC. Only XMT's will
                                1B39  4894 ; come thru here. Rcv's are complete via the BISYNC timer routine
                                1B39  4895 ;
                                1B39  4896 ;--
                                1B39  4897 FORK_DONE_BISYNC:
      51    00F4 D5    0F      1B39  4898 5$:     REMQUE  @UCB$Q_XG_POST(R5),R1           ; Get next buffer
                  3C    1D      1B3E  4899         BVS     40$                             ; If VS then none
        13    0A A1    91      1B40  4900         CMPB    XMTQ$B_BUFTYP(R1),S^#DYN$C_BUFIO ; If NEQ then something wrong
                  3A    12      1B44  4901         BNEQ    50$
                                1B46  4902
                                1B46  4903 ; This bit must be cleared to allow retransmits over the bisync protocol.
                                1B46  4904
                                1B46  4905         CLRBIT  #XG_DS_V_XMTING,UCB$W_DEVSTS(R5)
        53    0C A1    D0      1B4B  4906         MOVL    XMTQ$L_IRP(R1),R3               ; Get the IRP address
        50    1A A1    3C      1B4F  4907         MOVZWL  XMTQ$W_MSGSIZE(R1),R0           ; Get the message size
      50    50    10    78      1B53  4908         ASHL    #16,R0,R0                       ; Shift size into high word
                  07    12      1B57  4909         BNEQ    10$
      50    0054 8F    B0      1B59  4910         MOVW    #SS$_CTRLERR,R0                 ; Set error
                  08    11      1B5E  4911         BRB     20$                             ; Branch to complete the IO
      50    01    B0      1B60  4912 10$:    MOVW    S^#SS$_NORMAL,R0                ; Set successful status
    09 68 A5    0F    E4      1B63  4913         BBSC    #XG_DS_V_CLEAN,UCB$W_DEVSTS(R5),30$ ; If BS then clean outstanding
      38 A3    50    D0      1B68  4914 20$:    MOVL    R0,IRP$L_MEDIA(R3)              ; Set status
            012A    30      1B6C  4915         BSBW    TRANSMIT_IO_DONE                ; Complete the request
                  C8    11      1B6F  4916         BRB     5$                              ; Get next XMT to complete
      50    2C    B0      1B71  4917 30$:    MOVW    #SS$_ABORT,R0                   ; Set abort status
      54    00D0 C5    D0      1B74  4918         MOVL    UCB$A_XG_PRO_BUFFER(R5),R4      ; Get the protocol buffer
            F41E    30      1B79  4919         BSBW    CLEAN_FORK_ENTRY
      50    01    3C      1B7C  4920 40$:    MOVZWL  S^#SS$_NORMAL,R0
                  05      1B7F  4921         RSB
                          1B80  4922
                          1B80  4923 50$:    BUG_CHECK       NOBUFPCKT,FATAL
                          1B84  4924
```

```
                        1B84  4926          .SBTTL  RECEIVE_DONE - Complete a receive buffer
                        1B84  4927
                        1B84  4928 ;++
                        1B84  4929 ;RECEIVE_DONE
                        1B84  4930 ;
                        1B84  4931 ; FUNCTIONAL DESCRIPTION:
                        1B84  4932 ;
                        1B84  4933 ; When the board receives a buffer this routine is called to complete it. First
                        1B84  4934 ; a call is made to the protocol to strip off the header and use the
                        1B84  4935 ; appropriate information. If a non-fatal error is detected then the error is
                        1B84  4936 ; recorded in the protocol and the buffer os returned to the free buffer pool,
                        1B84  4937 ; and NOT given to the user. If no error occurs and the buffer is a protocol
                        1B84  4938 ; control message then the buffer is returned to the free buffer pool. if this
                        1B84  4939 ; is a data message and a IRP is free the buffer is completed with the IRP,
                        1B84  4940 ; else it is put on a queue to await I/O completion.
                        1B84  4941 ;
                        1B84  4942 ;       INPUTS: R2 = Message buffer address
                        1B84  4943 ;               R5 = UCB address
                        1B84  4944 ;
                        1B84  4945 ;       OUTPUTS R0 = Status
                        1B84  4946 ;               R2,R3,R5 are preserved
                        1B84  4947 ;--
                        1B84  4948 RECEIVE_DONE:                                        ; Receive done
          2C    BB      1B84  4949          PUSHR   #^M<R2,R3,R5>                       ; Save registers
     56   01    9A      1B86  4950          MOVZBL  #DLK$C_RCVMSG,R6                    ; Set that this is a RCV
          57    D4      1B89  4951          CLRL    R7                                  ; Clear error bits
     0708 8F    B3      1B8B  4952          BITW    #<XG$M_BCC_ERR!XG$M_RES_BIT_CNT>,- ; If EQL then no CRC error
       0E A2            1B8F  4953                  RCV_W_ERROR(R2)
          17    13      1B91  4954          BEQL    10$
  50    12 A2    3C      1B93  4955          MOVZWL  RCV_L_BACC+2(R2),R0                 ; Else check for header or data
  50    42 A5    A2      1B97  4956          SUBW2   UCB$W_DEVBUFSIZ(R5),R0              ; error
     50   06    91      1B9B  4957          CMPB    #XG$C_HEADER,R0                     ; If EQL then data crc error
          05    13      1B9E  4958          BEQL    5$
     57   02    A8      1BA0  4959          BISW    #DLK$M_HDRCRC,R7                    ; Set header CRC error
          12    11      1BA3  4960          BRB     15$
     57   04    A8      1BA5  4961 5$:      BISW    #DLK$M_DATACRC,R7                   ; Set data CRC error
          0D    11      1BA8  4962          BRB     15$
          06    E1      1BAA  4963 10$:     BBC     #XG$V_BUFOVR,-                      ; If BC then no buffer overrun
       08 0E A2            1BAC  4964                  RCV_W_ERROR(R2),15$
     57   20    A8      1BAF  4965          BISW    #DLK$M_RCVOVR,R7                    ; Else set rcv overrun
                        1BB2  4966          SETBIT  #XM$V_ERR_LOST,UCB$L_DEVDEPEND(R5) ; Set data lost error
  58    42 A2    DE      1BB7  4967 15$:     MOVAL   RCV_Z_HEADER(R2),R8                 ; Get buffer address
          59    D4      1BBB  4968          CLRL    R9                                  ; Clear reg
55   00D0 C5    D0      1BBD  4969          MOVL    UCB$A_XG_PRO_BUFFER(R5),R5          ; Set protocol buffer address
        E43B'   30      1BC2  4970          BSBW    DDCMP
          2C    BA      1BC5  4971          POPR    #^M<R2,R3,R5>                       ; Restore registers
  0C A2    59    B0      1BC7  4972          MOVW    R9,RCV_W_MSGSIZ(R2)                 ; Set transfer size
     56   09    91      1BCB  4973          CMPB    #DLK$C_ACTNOTCOM,R6                 ; If EQL then protl not active
          49    13      1BCE  4974          BEQL    40$
     57   0A    B3      1BD0  4975          BITW    #<DLK$M_PRSTERR!-                   ; If NEQ then fatal protocol
                        1BD3  4976                  DLK$M_HDRERR>,R7                    ; error go to shutdown the
          57    12      1BD3  4977          BNEQ    55$                                 ; circuit
57   00C0 8F    B3      1BD5  4978          BITW    #<DLK$M_STRTRCV!-                   ; If NEQ then trib error
                        1BDA  4979                  DLK$M_MNTRCV>,R7                    ; go to shutdown the circuit
          57    12      1BDA  4980          BNEQ    60$
  14 57    0E    E1      1BDC  4981          BBC     #DLK$V_TRNLK,R7,20$                 ; If BC don't turn link
          01    AA      1BE0  4982          BICW    #XG_DS_M_XMTING,-                   ; Start transmitter
```

```
                68 A5          1BE2  4983                    UCB$W_DEVSTS(R5)
            0084 8F    BB      1BE4  4984          PUSHR     #^M<R2,R7>                  ; Save the register
                 51    D4      1BE8  4985          CLRL      R1                          ; Set no status for start xmt
                   FOBC  30     1BEA  4986          BSBW      START_TRANSMIT              ; Start the tranmsit
            0084 8F    BA      1BED  4987          POPR      #^M<R2,R7>                  ; Restore the register
              5A 50    E9      1BF1  4988          BLBC      R0,70$                      ; Branch on error
        08 57    0A    E1      1BF4  4989 20$:     BBC       #DLK$V_XMTCMP,R7,23$        ; If BC no transmit to complete
                 52    DD      1BF8  4990          PUSHL     R2                          ; Save receive buffer
                   00F8  30     1BFA  4991          BSBW      FINISH_XMT_IO               ; Else branch to post the I/O
                 52 8ED0  1BFD  4992          POPL      R2                          ; Restore receive buffer
        15 57    08    E1      1C00  4993 23$:     BBC       #DLK$V_RCVACK,R7,40$        ; If BC then not a data message
                               1C04  4994
    53    00DC D5    0F      1C04  4995          REMQUE    @UCB$Q_XG_RCVS(R5),R3       ; Remov waiting rcv I/O request
                 02    1D      1C09  4996          BVS       25$                         ; VS then no packet to complete
                 47    11      1C0B  4997          BRB       FINISH_RCV_IO               ; If found then finish the I/O
                               1C0D  4998
00D8 D5    62    0E      1C0D  4999 25$:     INSQUE    (R2),@UCB$Q_XG_ATTN+4(R5)   ; Else, queue message buffer
            0446   30     1C12  5000 30$:     BSBW      POKE_USER                   ; Poke the user
      50    01     3C     1C15  5001          MOVZWL    S^#SS$_NORMAL,R0            ; Set normal return
                 05      1C18  5002          RSB
                               1C19  5003
00E8 D5    62    0E      1C19  5004 40$:     INSQUE    (R2),@UCB$Q_XG_FREE+4(R5)   ; Return the buff to free list
                         1C1E  5005          DSBINT    UCB$B_DIPL(R5)              ; Disable interrupts
            FADA   30     1C25  5006          BSBW      START_RECEIVE               ; Start any receives
                         1C28  5007          ENBINT                                ; Enable interrupts
                 05      1C2B  5008 50$:     RSB
                               1C2C  5009
                               1C2C  5010 55$:     SETBIT    #XM$V_ERR_FATAL,-           ; Set that a fatal error occurd
                               1C2C  5011                    UCB$L_DEVDEPEND(R5)
                               1C31  5012
                               1C31  5013          .IF       DF ERR$$$
                               1C31  5014          incw      ucb$w_xg_prst(r5)           ; inc persitant error count
                               1C31  5015          .ENDC     ;DF ERR$$$
                               1C31  5016
                   16    11      1C31  5017          BRB       65$
                         C8    1C33  5018 60$:     BISL      #<XM$M_ERR_START!-          ; Assume the trib error occurrd
                               1C34  5019                    XM$M_ERR_TRIB>,-            ; because a STRT was received
 44 A5    00C00000 8F            1C34  5020                    UCB$L_DEVDEPEND(R5)
                               1C3B  5021
                               1C3B  5022          .IF       DF ERR$$$
                               1C3B  5023          incw      ucb$w_xg_strt(r5)           ; inc start rcv'd error count
                               1C3B  5024          .ENDC     ;DF ERR$$$
                               1C3B  5025
          0A 57    07    E1      1C3B  5026          BBC       #DLK$V_MNTRCV,R7,65$        ; If BC then true
                               1C3F  5027          CLRBIT    #XM$V_ERR_START,UCB$L_DEVDEPEND(R5)
                               1C44  5028          SETBIT    #XM$V_ERR_MAINT,UCB$L_DEVDEPEND(R5) ; Else set maint msg rcv'd
      50    20D4 8F    3C      1C49  5029 65$:     MOVZWL    #SS$_DEVINACT,R0            ; Set protocol inactive
      00E8 D5    62    0E      1C4E  5030 70$:     INSQUE    (R2),@UCB$Q_XG_FREE+4(R5)   ; Return the buff to free list
                         05    1C53  5031          RSB
                               1C54  5032
```

```
                                1C54  5034          .SBTTL  FINISH_RCV_IO - Finish receive I/O processing
                                1C54  5035 ;++
                                1C54  5036 ; FINISH_RCV_IO - Finish receive I/O processing
                                1C54  5037 ;
                                1C54  5038 ; FUNCTIONAL DESCRIPTION:
                                1C54  5039 ;
                                1C54  5040 ; This routine completes a receive operation that has been matched with a
                                1C54  5041 ; message block. After the receive has been completed the message free list is
                                1C54  5042 ; filled and a receive is started if needed.
                                1C54  5043 ;
                                1C54  5044 ; INPUTS:
                                1C54  5045 ;
                                1C54  5046 ;       R2 = message buffer address
                                1C54  5047 ;          = 0 if I/O is being aborted
                                1C54  5048 ;       R3 = I/O packet address
                                1C54  5049 ;       R5 = UCB address
                                1C54  5050 ;
                                1C54  5051 ;       IPL = FIPL
                                1C54  5052 ;
                                1C54  5053 ;       For FINISH_RCV_IO_LAPB (R2) = address of data
                                1C54  5054 ;                              R0  = transfer size of message
                                1C54  5055 ;
                                1C54  5056 ; OUTPUTS:
                                1C54  5057 ;
                                1C54  5058 ;       R5 = UCB address
                                1C54  5059 ;
                                1C54  5060 ;       The request is completed via I/O post.
                                1C54  5061 ;--
                                1C54  5062 FINISH_RCV_IO:                               ; Finish receive I/O request
   62    48 A2    DE           1C54  5063          MOVAL   RCV_T_DATA(R2),(R2)         ; Insert address of the data
   50    0C A2    3C           1C58  5064          MOVZWL  RCV_W_MSGSIZ(R2),R0         ; Get size of transfer
                                1C5C  5065 FINISH_RCV_IO_LAPB:
                                1C5C  5066 FINISH_RCV_IO_BISYNC:
         51    D4               1C5C  5067          CLRL    R1                          ; Assume error
   2C A3    52    D0           1C5E  5068          MOVL    R2,IRP$L_SVAPTE(R3)         ; Save block address
04 A2    38 A3    D0           1C62  5069          MOVL    IRP$L_MEDIA(R3),4(R2)       ; Insert saved user VA
      42 A5    A0               1C67  5070          ADDW    UCB$W_DEVBUFSIZ(R5),-       ; Adjust unit quota
      010C C5                   1C6A  5071                  UCB$W_XG_QUOTA(R5)
   32 A3    50    B1           1C6D  5072          CMPW    R0,IRP$W_BCNT(R3)           ; Request larger than actual?
         04    1B               1C71  5073          BLEQU   20$                         ; Br if no
   50    32 A3    3C           1C73  5074          MOVZWL  IRP$W_BCNT(R3),R0           ; Set size to min. of two sizes
   32 A3    50    B0           1C77  5075 20$:     MOVW    R0,IRP$W_BCNT(R3)           ; Set size to transfer
50    50    10    78           1C7B  5076          ASHL    #16,R0,R0                   ; Set up status
         07    12               1C7F  5077          BNEQ    25$                         ; Br if success
50    0054 8F    B0           1C81  5078          MOVW    #SS$_CTRLERR,R0             ; Set data path error
         03    11               1C86  5079          BRB     30$
      50    01    B0           1C88  5080 25$:     MOVW    S^#SS$_NORMAL,R0            ; Set success
   38 A3    50    D0           1C8B  5081 30$:     MOVL    R0,IRP$L_MEDIA(R3)          ; Set status and size
                                1C8F  5082
   0A A2    13    90           1C8F  5083          MOVB    #DYN$C_BUFIO,IRP$B_TYPE(R2) ; PSI expects this in all buffs
         16    10               1C93  5084          BSBB    IO_DONE                     ; Post the I/O request
         F987    30             1C95  5085          BSBW    FICLFREELIST                ; Return rcv buffer to free q
         05    05               1C98  5086          RSB                                 ; and start next RCV
                                1C99  5087
                                1C99  5088 ;
                                1C99  5089 ; Complete a transfer I/O operation
                                1C99  5090 ;
```

```
                          1C99  5091 TRANSMIT_IO_DONE:
                07   E1   1C99  5092          BBC     #XMTQ$V_INTERNAL,-                  ; If BC then not an "Interal"
             0D 1F   A1        1C9B  5093                  XMTQ$B_FLAG(R1),IO_DONE             ; IRP, else must dealloc the
                          1C9E  5094                                                      ; buffer used to transmit
                28   BB   1C9E  5095          PUSHR   #^M<R3,R5>                          ; Save registers
              50    51   D0   1CA0  5096          MOVL    R1,R0                               ; Set to dealloc the buffer
        00000000'GF   16   1CA3  5097          JSB     G^COM$DRVDEALMEM                    ; Deall the buffer
                28   BA   1CA9  5098          POPR    #^M<R3,R5>                          ; Restore registers
                          1CAB  5099 IO_DONE:                                             ; Comp a transfer I/O operation
                44   A5   D0   1CAB  5100          MOVL    UCB$L_DEVDEPEND(R5),-               ; Set other info
                3C   A3        1CAE  5101                  IRP$L_MEDIA+4(R3)
                          1CB0  5102
                          1CB0  5103 ; Clear bit because it has been reported to the user
                          1CB0  5104
                          1CB0  5105          CLRBIT  #XM$V_STS_ORUN,UCB$L_DEVDEPEND(R5)
                07   E1   1CB5  5106          BBC     #IRP$V_DIAGBUF,-                    ; Br if no diagnostic buffer
             14 2A   A3        1CB7  5107                  IRP$W_STS(R3),20$
          50   4C B3   08   C1  1CBA  5108          ADDL3   #8,@IRP$L_DIAGBUF(R3),R0            ; Addr buffer past start time
        80   00000000'GF   7D   1CBF  5109          MOVQ    G^EXE$GQ_SYSTIME,(R0)+              ; Insert stop time
           80    0082 C5   3C   1CC6  5110          MOVZWL  UCB$W_ERRCNT(R5),(R0)+              ; Insert error counter
                  0374   30   1CCB  5111          BSBW    REGDUMP
             00000000'GF   17   1CCE  5112 20$:    JMP     G^COM$POST                          ; Post the I/O
                          1CD4  5113
```

```
                     1CD4  5115
                     1CD4  5116          .SBTTL  TRANSMIT_DONE - Transmit completion routine
                     1CD4  5117 ;++
                     1CD4  5118 ; TRANSMIT_DONE - Transmit completion routine
                     1CD4  5119 ;
                     1CD4  5120 ; FUNCTIONAL DESCRIPTION:
                     1CD4  5121 ;
                     1CD4  5122 ; This routine is called when a transmit buffer needs completion. If the
                     1CD4  5123 ; transmit buffer completed was a protocol control buffer then nothing
                     1CD4  5124 ; happens with the buffer. In fact it should be part of a permantly allocated
                     1CD4  5125 ; structure such as the UCB. If the transmit was a data buffer, the routine
                     1CD4  5126 ; calls the protocol to deal with the buffer. If the protocol notifies the
                     1CD4  5127 ; driver that it has XMT's to send to I/O completion, the driver pulls these
                     1CD4  5128 ; buffers off the complete queue and sends them off to complete via COM$POST.
                     1CD4  5129 ;
                     1CD4  5130 ; INPUTS:
                     1CD4  5131 ;       R2 = Address of buffer to complete
                     1CD4  5132 ;       R3 = If error then contains the error from the device
                     1CD4  5133 ;       R5 = UCB address
                     1CD4  5134 ;
                     1CD4  5135 ;       IPL = FIPL
                     1CD4  5136 ;
                     1CD4  5137 ; OUTPUTS:
                     1CD4  5138 ;
                     1CD4  5139 ;       R5 = UCB address
                     1CD4  5140 ;
                     1CD4  5141 ;--
                     1CD4  5142 TRANSMIT_DONE:
     58   52   D0    1CD4  5143          MOVL    R2,R8                           ; Put buffer addr into R8
          2C   BB    1CD7  5144          PUSHR   #^M<R2,R3,R5>                   ; Save registers
55   00D0 C5   D0    1CD9  5145          MOVL    UCB$A_XG_PRO_BUFFER(R5),R5      ; Set protocol buffer address
     56   02   9A    1CDE  5146          MOVZBL  #DLK$C_XMTMSG,R6                ; Set up to put on RTOQ if the
     57   01   9A    1CE1  5147          MOVZBL  #DLK$M_MSGSENT,R7               ; msg needs to be timed out
          59   D4    1CE4  5148          CLRL    R9
        E317'  30    1CE6  5149          BSBW    DDCMP                           ; Branch to protocol
          2C   BA    1CE9  5150          POPR    #^M<R2,R3,R5>                   ; Restore registers
  02 57   0A   E1    1CEB  5151          BBC     #DLK$V_XMTCMP,R7,30$            ; If BC then no XMT's to compl
          04   10    1CEF  5152          BSBB    FINISH_XMT_IO                   ; Else complete the XMT
     50   01   3C    1CF1  5153 30$:     MOVZWL  S^#SS$_NORMAL,R0                ; Set normal return
               05    1CF4  5154          RSB
                     1CF5  5155
                     1CF5  5156 FINISH_XMT_IO:
     03C0 8F   BB    1CF5  5157          PUSHR   #^M<R6,R7,R8,R9>                ; Save registers
56   00D0 C5   D0    1CF9  5158          MOVL    UCB$A_XG_PRO_BUFFER(R5),R6      ; Get protocol buffer address
  58   28 B6   0F    1CFE  5159 5$:      REMQUE  @TF$Q_CMPQ(R6),R8               ; Get the next entry on queue
          27   1D    1D02  5160          BVS     20$                             ; If VS then branch to finish
  53   0C A8   D0    1D04  5161          MOVL    XMTQ$L_IRP(R8),R3               ; Get IRP associated with XMT
  50   1A A8   3C    1D08  5162          MOVZWL  XMTQ$W_MSGSIZE(R8),R0           ; Get transfer size
     50   06   A2    1D0C  5163          SUBW2   #XG$C_READER,R0                 ; Subtract out the protocol hdr
50   50   10   78    1D0F  5164          ASHL    #16,R0,R0                       ; Set up status
          07   12    1D13  5165          BNEQ    10$                             ; Br if data transmitted
50   0054 8F   B0    1D15  5166          MOVW    #SS$_CTRLERR,R0                 ; Set device error
          03   11    1D1A  5167          BRB     15$
     50   01   B0    1D1C  5168 10$:     MOVW    #SS$_NORMAL,R0                  ; Assume success
  38 A3   50   D0    1D1F  5169 15$:     MOVL    R0,IRP$L_MEDIA(R3)
     51   58   D0    1D23  5170          MOVL    R8,R1                           ; Set XMTQ buff address
        FF70   30    1D26  5171          BSBW    TRANSMIT_IO_DONE                ; Complete the IO
```

```
                                                        ; Edit 11
                D3   11  1D29  5172                BRB     5$                          ; Get next to complete
        03C0    8F   BA  1D2B  5173 20$:           POPR    #^M<R6,R7,R8,R9>            ; Restore registers
                     05  1D2F  5174                RSB
```

```
                                1D30  5176          .SBTTL  TIMER - CTS and Device timer wakeup routines
                                1D30  5177
                                1D30  5178 ;++
                                1D30  5179 ;CTS_TIMER - CTS wakeup routine
                                1D30  5180 ;
                                1D30  5181 ; This routine is called to check if CTS has come high
                                1D30  5182 ;
                                1D30  5183 ; Inputs:
                                1D30  5184 ;                   R5 = Address of CTS TQE entry in UCB
                                1D30  5185 ;
                                1D30  5186 ;--
                                1D30  5187 CTS_TIMER:
                     03FF 8F    BB  1D30  5188          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Save the registers
              55  00000094 8F    C2  1D34  5189          SUBL    #UCB$L_XG_TQE,R5                   ; Point to start of UCB
                                1D3B  5190          DSBINT  UCB$B_FIPL(R5)                     ; Sync access to unit
              68 A5    4000 8F    AA  1D42  5191          BICW2   #XG_DS_M_CTSTQE_RUN,UCB$W_DEVSTS(R5) ; Set timer not q'd
                22 68 A5    02    E1  1D48  5192          BBC     #XG_DS_V_INITED,UCB$W_DEVSTS(R5),20$ ; If BC then device not active
                     0120 C5    95  1D4D  5193          TSTB    UCB$B_XG_XSTATE(R5)                ; If EQL then idle
                          1C    13  1D51  5194          BEQL    20$
                    17 0120 C5    E8  1D53  5195          BLBS    UCB$B_XG_XSTATE(R5),20$            ; If set then XMT'r is "ON"
                 0120 C5    06    91  1D58  5196          CMPB    #XG$C_DRPCTS,UCB$B_XG_XSTATE(R5) ; If NEQ then not drop
                          06    12  1D5D  5197          BNEQ    5$                                 ; CTS state
                     0120 C5    94  1D5F  5198          CLRB    UCB$B_XG_XSTATE(R5)                ; Set up to see if we can
                          05    11  1D63  5199          BRB     10$                                ; transmit and try to start up
                                1D65  5200 5$:
                                1D65  5201          .IF     DF CTS$$$
                                1D65  5202          movzbl  ucb$b_cts_last(r5),r0              ; set slot to incr
                                1D65  5203          incb    ucb$b_cts_buf(r5)[r0]
                                1D65  5204          .ENDC                                      ; DF CTS$$$ end
                                1D65  5205
              0120 C5    04    90  1D65  5206          MOVB    #XG$C_SWFCTS,UCB$B_XG_XSTATE(R5) ; Short WCTS state
                    51    D4  1D6A  5207 10$:     CLRL    R1                                 ; Set no status for start xmt
                 0086    30  1D6C  5208          BSBW    DEVTIMER_ALT                       ; Restart transmit
                                1D6F  5209 20$:     ENBINT                                     ; Reset IPL
                03FF 8F    BA  1D72  5210          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Restore registers
                          05    1D76  5211          RSB
                                1D77  5212
                                1D77  5213
                                1D77  5214 ;++
                                1D77  5215 ; DEVTIMER - Device wakeup routine
                                1D77  5216 ;
                                1D77  5217 ; Functional Description:
                                1D77  5218 ;
                                1D77  5219 ; This routine is called when the protocol timer every second when the
                                1D77  5220 ; protocol timer ticks to start any transmits the protocol may
                                1D77  5221 ; have queued as a result of the timer going off. It also checks for CTS coming
                                1D77  5222 ; high as a result of RTS being asserted.
                                1D77  5223 ;
                                1D77  5224 ;       INPUTS:
                                1D77  5225 ;               R1 = Error status from protocol specific timer routine
                                1D77  5226 ;               R5 = Address of the UCB
                                1D77  5227 ;
                                1D77  5228 ;
                                1D77  5229 ;       OUTPUTS:
                                1D77  5230 ;               R0 = Status of line is LBS then line is up
                                1D77  5231 ;                                    LBC then line is down
                                1D77  5232 ;               R1 = Return error status from driver
```

```
                                1D77  5233 ;               On error the line is shut down
                                1D77  5234 ;
                                1D77  5235          .ENABL  LSB
                                1D77  5236 DEVTIMER:
                                1D77  5237
                                1D77  5238 ; First check to see if the timer on the xmt inprogress has expired
                                1D77  5239 ; If it has then we will increment the error counter for the device
                                1D77  5240 ; and shutdown the board.
                                1D77  5241
                                1D77  5242          DSBINT  UCB$B_DIPL(R5)
     52   0104 C5    D0        1D7E  5243          MOVL    UCB$L_XG_XMTEND(R5),R2          ; If EQL no xmt inporgress
                16   13        1D83  5244          BEQL    2$
50   00000000'GF    D0        1D85  5245          MOVL    G^EXE$GL_ABSTIM,R0              ; Get time
          52   50   D1        1D8C  5246          CMPL    R0,R2                           ; If LEQ then no expired
               0A   15        1D8F  5247          BLEQ    2$                              ;
                               1D91  5248          ENBINT
          0082 C5   B6        1D94  5249          INCW    UCB$W_ERRCNT(R5)                ; Incr the error counter
             0072   31        1D98  5250          BRW     30$                             ; Go to shut down the device
                               1D9B  5251
                               1D9B  5252 2$:      ENBINT
   6A 44 A5   10   E0        1D9E  5253          BBS     #XM$V_ERR_FATAL,UCB$L_DEVDEPEND(R5),30$ ; If BS then fatal err
      66 51   01   E0        1DA3  5254          BBS     #DLK$V_PRSTERR,R1,30$           ; If BS then psrt err shut down
                               1DA7  5255                                              ; the device
          0120 C5   95        1DA7  5256          TSTB    UCB$B_XG_XSTATE(R5)             ; If NEQ then transmitter going
               1A   12        1DAB  5257          BNEQ    10$
                               1DAD  5258 ;
                               1DAD  5259 ; If the protocol timer has expired then clear the XMTING flag so that
                               1DAD  5260 ; the control message which must be sent can be. This is true when the
                               1DAD  5261 ; device is running half duplex. If the message with the select flag is dropped
                               1DAD  5262 ; then we must be sure that we can send anouthe message with a select flag.
                               1DAD  5263 ;
    05 51   0F   E1            1DAD  5264          BBC     #DLK$V_TMREXPD,R1,5$
                               1DB1  5265          CLRBIT  #XG_DS_V_XMTING,UCB$W_DEVSTS(R5)
          51   DD              1DB6  5266 5$:      PUSHL   R1
        EEEE   30              1DB8  5267          BSBW    START_TRANSMIT                  ; Else try to start a transmit
          51 8ED0              1DBB  5268          POPL    R1
        4C 50   E9             1DBE  5269          BLBC    R0,30$                          ; If LBC then fatal error
51   8000 8F   AA              1DC1  5270          BICW    #DLK$M_TMREXPD,R1               ; Clear timer expired
                05             1DC6  5271          RSB
                               1DC7  5272
     50   01   D0              1DC7  5273 10$:     MOVL    #SS$_NORMAL,R0                  ; Assume device is ok.
   2C 0120 C5   E8             1DCA  5274          BLBS    UCB$B_XG_XSTATE(R5),15$         ; If LBC then XMTer is on
27 68 A5   0E   E0             1DCF  5275          BBS     #XG_DS_V_CTSTQE_RUN,UCB$W_DEVSTS(R5),15$ ; Br if CTS timer is
                               1DD4  5276                                              ;   running
                               1DD4  5277          DSBINT  UCB$B_DIPL(R5)                  ; Disable interr
   51   24 A5   D0             1DDB  5278          MOVL    UCB$L_CRB(R5),R1                ; Get the CRB address
   54   2C B1   D0             1DDF  5279          MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R4    ; Get the CSR
   04 A4   04   B0             1DE3  5280          MOVW    #XG$C_MODEM,XG$C_MISC_REG(R4)   ; Set to get modem IR
   06 A4   10   B3             1DE7  5281          BITW    #XG$M_CTS,XG$C_IND_ADDR(R4)     ; Has CTS been detected
                               1DEB  5282          ENBINT                                  ; Enable interrupts
             0C   13           1DEE  5283          BEQL    20$                             ; If EQL then not detected
             01   90           1DF0  5284          MOVB    #XG$C_XMTING,-                  ; Set state to XMTing
        0120 C5                1DF2  5285                  UCB$B_XG_XSTATE(R5)
                               1DF5  5286
                               1DF5  5287 DEVTIMER_ALT:
          EEB1   30            1DF5  5288          BSBW    START_TRANSMIT                  ; Start the transmit
        12 50   E9             1DF8  5289          BLBC    R0,30$                          ; If LBC then problem
```

```
                05  1DFB  5290 15$:    RSB
                    1DFC  5291
          07    E0  1DFC  5292 20$:    BBS     #XM$V_CHR_TRIB,-                    ; If BS then trib mode don't
 06 44 A5             1DFE  5293             UCB$L_DEVDEPEND(R5),22$             ; time the return of CTS
  0121 C5   97  1E01  5294             DECB    UCB$B_XG_WFCTS_SEC(R5)              ; If EQL then device hasn't
          01    13  1E05  5295             BEQL    25$                                 ; repsonded in a reasonable tim
                05  1E07  5296 22$:    RSB
                    1E08  5297
                    1E08  5298 25$:    SETBIT  #XM$V_STS_DISC,UCB$L_DEVDEPEND(R5)  ; Set modem disconnect sts
                    1E0D  5299
                    1E0D  5300         .IF     DF ERR$$$
                    1E0D  5301         incw    ucb$w_xg_disc(r5)                   ; inc disconect modem err count
                    1E0D  5302         .ENDC   ;DF ERR$$$
                    1E0D  5303
                    1E0D  5304 ;
                    1E0D  5305 ; TFB/GFB blocks are deallocated on shutdown. Make sure R0 is
                    1E0D  5306 ; clear so that the timer routine does not try to access them.
                    1E0D  5307 ;
                    1E0D  5308 30$:    SETBIT  #XM$V_ERR_FATAL,UCB$L_DEVDEPEND(R5) ; Set fatal error
        02E9    30  1E12  5309         BSBW    SHUTDOWN_LINE                       ; Shut down the device
          50    D4  1E15  5310         CLRL    R0
                05  1E17  5311         RSB
                    1E18  5312         .DSABL  LSB
                    1E18  5313
```

```
                                        .SBTTL  BISYNC_TIMER - Bisync timer
                                ;++
                                ; BISYNC_TIMER - Bisync mode timer routine
                                ;
                                ; This is the timer routine for BISYNC operation. Each time the clock ticks,
                                ; it calls this routine to check first if a XMT is waiting for CTS to come
                                ; high. Second to see if any characters were received by the RCVer. If
                                ; characters were received then it will call the framming routine to
                                ; find out what to do with the character. That is how the character gets put
                                ; into the actual receive buffer that will get posted to the  user.
                                ; The following is a diagram of the possible status the framing
                                ; routine may retrun.
                                ;
                                ; The following indicates bits returned in R0 from the framming routine:
                                ;
                                ; BIT 0: BCHAR = Buffer the character
                                ; BIT 1: BCHRP = Buffer the character in the previous position
                                ; BIT 2: CMPBF  = Complete the receive to the user
                                ;
                                ;       !--CMPBF--!--BCHRP---!--BCHAR--!
                                ;       !---------------------------------
                                ;       !   0     !    0     !    0     ! Buffer the character in next position
                                ;       !   0     !    0     !    1     ! Ignore the character
                                ;       !   0     !    1     !    0     ! Illegal state
                                ;       !   0     !    1     !    1     ! Buffer the character in previous position
                                ;       !   1     !    0     !    0     ! Buffer the character in the next postion and
                                ;       !   1     !    0     !    1     ! Ignore the character and complete the read
                                ;       !   1     !    1     !    0     ! Illegal state
                                ;       !   1     !    1     !    1     ! Buffer character in previous position and co
                                ;       ! ---------------------------
                                ;
                                ;       INPUTS
                                ;               R5 = TQE address
                                ;               IPL = timer IPL
                                ;
                                ;       OUTPUTS
                                ;               All regs preserved.
                                ;               IRP may be completed VIA IOPOST
                                ;--
                                BISYNC_TIMER:
    03FC 8F    BB   1E18  5356          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9>
54    14 A5    D0   1E1C  5357          MOVL    TQE$L_FR4(R5),R4                ; Set R4 with protocol buffer addres
55    10 A5    D0   1E20  5358          MOVL    TQE$L_FR3(R5),R5                ; Set R5 with UCB address
    0120 C5    95   1E24  5359          TSTB    UCB$B_XG_XSTATE(R5)             ; If EQL XMTer is idle
         03    13   1E28  5360          BEQL    CHECK_RCV
       017A    31   1E2A  5361          BRW     CHECK_XMT                       ; Else go see what the XMTr is
                    1E2D  5362                                                  ; waiting for
                    1E2D  5363 CHECK_RCV:
         54    DD   1E2D  5364          PUSHL   R4                              ; Save R4 addr of protocol buffer
51    24 A5    D0   1E2F  5365          MOVL    UCB$L_CRB(R5),R1                ; Get CRB address
54    2C B1    D0   1E33  5366          MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R4    ; Get device CSR
                    1E37  5367          DSBINT  UCB$B_DIPL(R5)                  ; Sync to DIPL to read device regs
64  4000 8F    B3   1E3E  5368          BITW    #XG$M_ERROR,XG$C_RCV_CSR(R4)
         08    13   1E43  5369          BEQL    5$
                    1E45  5370          ENBINT
         54 8ED0   1E48  5371          POPL    R4
```

```
                    65   11   1E4B  5372          BRB     RCV_ERROR
                                1E4D  5373 5$:     ENBINT
          4E 68 A5    06   E1   1E50  5374          BBC     #XG_DS_V_RCV_DONEP,UCB$W_DEVSTS(R5),10$ ; BC buff not queued
           52   00F2 C5    3C   1E55  5375          MOVZWL  UCB$W_XG_RCV_INIT(R5),R2        ; Get count of # of bytes to input
                                1E5A  5376          DSBINT  UCB$B_DIPL(R5)                  ; Sync to DIPL to read device regs
             04 A4    07   B0   1E61  5377          MOVW    #XG$C_PRI_RCV1,XG$C_MISC_REG(R4) ; Set char count reg to read
             53   06 A4    3C   1E65  5378          MOVZWL  XG$C_IND_ADDR(R4),R3            ; Get the count
                      3E    13   1E69  5379          BEQL    30$                             ; Br EQL no more chars to rcv
                                1E6B  5380          ENBINT                                  ; Reset interrupts
                         54 8ED0  1E6E  5381          POPL    R4                              ; Restore protocol buffer address
       53   02   0F    00   F0   1E71  5382          INSV    #0,#15,#2,R3                    ; Clear high order bits
                52    53   A2   1E76  5383          SUBW2   R3,R2                           ; Get number of char input since las
                      2B    13   1E79  5384          BEQL    20$                             ; If eql then none
             58   00EC C5    D0   1E7B  5385          MOVL    UCB$L_XG_RCV_INPR(R5),R8        ; Get buffer to move rcv data to
                      24    13   1E80  5386          BEQL    20$                             ; If EQL then buffer complete
               58   42 A8    DE   1E82  5387          MOVAL   RCV_Z_HEADER(R8),R8             ; Get the start of the data
             59   00F0 C5    3C   1E86  5388          MOVZWL  UCB$W_XG_RCV_INPR_INDX(R5),R9   ; Get the index into the buffer
             00F2 C5    53   B0   1E8B  5389          MOVW    R3,UCB$W_XG_RCV_INIT(R5)        ; Update saved count
               56   1C A4    D0   1E90  5390          MOVL    BISYNC$A_RCV_BUFFER(R4),R6      ; Get address of buffer given to dev
               56   42 A6    DE   1E94  5391          MOVAL   RCV_Z_HEADER(R6),R6             ; Get address of data
               57   0C A4    3C   1E98  5392          MOVZWL  BISYNC$W_RCV_INDEX(R4),R7       ; Get the index into that buffer
                                1E9C  5393          CLRBIT  #BISYNC$V_RCV_COMPLETE,BISYNC$W_STATUS(R4)
                      2A    11   1EA1  5394          BRB     CHECK_NEXT
                                1EA3  5395
                         54 8ED0  1EA3  5396 10$:     POPL    R4
                   00C7    31   1EA6  5397 20$:     BRW     TIMER_END
                                1EA9  5398
                                1EA9  5399 30$:     ENBINT                                  ; Reset interrupts
                         54 8ED0  1EAC  5400          POPL    R4                              ; Restore protocol buffer address
                   00C3    31   1EAF  5401          BRW     TIMER_END_CHKCMP
                                1EB2  5402
                                1EB2  5403 RCV_ERROR:
                                1EB2  5404          SETBIT  #XM$V_STS_ORUN,UCB$L_DEVDEPEND(R5)
       53   00DC D5    0F   1EB7  5405          REMQUE  @UCB$Q_XG_RCVS(R5),R3           ; If there is an rcv posted
                      0C    1D   1EBC  5406          BVS     10$                             ; complete it in error
             38 A3    2C   3C   1EBE  5407          MOVZWL  #SS$_ABORT,IRP$L_MEDIA(R3)
                      54    DD   1EC2  5408          PUSHL   R4
                   FDE4    30   1EC4  5409          BSBW    IO_DONE
                         54 8ED0  1EC7  5410          POPL    R4
                   00AD    31   1ECA  5411 10$:     BRW     TIMER_END_CHKCMP_ALT
                                1ECD  5412
                                1ECD  5413 CHECK_NEXT:
             51   6647    9A   1ECD  5414          MOVZBL  (R6)[R7],R1                     ; Get next char
                      57    B6   1ED1  5415          INCW    R7                              ; Incr the index
       50   015C C5    DE   1ED3  5416          MOVAL   UCB$Q_XG_STATE_INFO(R5),R0      ; Pass address of state info
            0158 D5    16   1ED8  5417          JSB     @UCB$A_XG_FRAME_ADDR(R5)        ; Branch to the framing routine
                                1EDC  5418
                                1EDC  5419          ASSUME  XG$V_BUFFER_CHAR EQ 0
                                1EDC  5420          ASSUME  XG$V_BUFFER_IN_PREV_POS EQ 1
                                1EDC  5421          ASSUME  XG$V_COMPLETE_READ EQ 2
                                1EDC  5422
                                1EDC  5423          CASE    R0,TYPE=B,<-                    ; Case according to result in R0
                                1EDC  5424                  BCHAR_NPOS,-                    ; Buffer char in next position
                                1EDC  5425                  IGN_CHAR,-                      ; Ignore character
                                1EDC  5426                  ILLEGAL,-                       ; Illegal state
                                1EDC  5427                  BCHAR_PPOS,-                    ; Buffer char in prev position
                                1EDC  5428                  BCHAR_NPOS_CMP,-                ; Buffer char in nxt pos and cmp
```

```
                     1EDC  5429                IGN_CHAR_CMP,-                  ; Ignore char and complete read
                     1EDC  5430                ILLEGAL,-                       ; Illegal state
                     1EDC  5431                BCHAR_PPOS_CMP>                 ; Buffer char in prev pos and cmp re
                     1EF0  5432
                     1EF0  5433 ILLEGAL:       BUG_CHECK       NOBUFPCKT,FATAL
                     1EF4  5434
                     1EF4  5435 ;
                     1EF4  5436 ; For the following routines please remember:
                     1EF4  5437 ;
                     1EF4  5438 ;       R1 = Character to buffer
                     1EF4  5439 ;       R4 = Protocol buffer address
                     1EF4  5440 ;       R5 = UCB address
                     1EF4  5441 ;       R6 = Address of rcv buffer given to board
                     1EF4  5442 ;       R7 = Index into above buffer
                     1EF4  5443 ;       R8 = Address of rcv buffer which will be completed to the user
                     1EF4  5444 ;       R9 = Index into above buffer
                     1EF4  5445 ;
                     1EF4  5446
                     1EF4  5447 BCHAR_NPOS:
     6849    51   90 1EF4  5448          MOVB    R1,(R8)[R9]                     ; Mov byte into position
             59   B6 1EF8  5449          INCW    R9                              ; Incr the index
                     1EFA  5450                                                  ; Fall thru to do next char
                     1EFA  5451 IGN_CHAR:
        D0 52   F5 1EFA  5452          SOBGTR  R2,CHECK_NEXT                   ; Branch to check next char
             63   11 1EFD  5453          BRB     TIMER_END_UPD                   ; If out of chars go to timer end
                     1EFF  5454
                     1EFF  5455 BCHAR_PPOS:
  50   59   01   C3 1EFF  5456          SUBL3   #1,R9,R0                        ; Set index to previous position
     6840    51   90 1F03  5457          MOVB    R1,(R8)[R0]                     ; Buffer the character
        C3 52   F5 1F07  5458          SOBGTR  R2,CHECK_NEXT                   ; Branch to check the next char
             56   11 1F0A  5459          BRB     TIMER_END_UPD                   ; If out of chars go to timer end
                     1F0C  5460
                     1F0C  5461 BCHAR_NPOS_CMP:
     6849    51   90 1F0C  5462          MOVB    R1,(R8)[R9]                     ; Buffer char in position
             59   B6 1F10  5463          INCW    R9                              ; Incr the index
             52   B7 1F12  5464          DECW    R2                              ; Decr in count
                     1F14  5465                                                  ; Fall thru to jump to complete
                     1F14  5466 IGN_CHAR_CMP:
             0A   11 1F14  5467          BRB     TIMER_END_COMP
                     1F16  5468
                     1F16  5469 BCHAR_PPOS_CMP:
  50   59   01   C3 1F16  5470          SUBL3   #1,R9,R0                        ; Set the index to the previous pos
     6840    51   90 1F1A  5471          MOVB    R1,(R8)[R0]                     ; Buffer char
             52   B7 1F1E  5472          DECW    R2                              ; Decr char count
                     1F20  5473                                                  ; Fall thru to complete the message
                     1F20  5474 TIMER_END_COMP:
                     1F20  5475          SETBIT  #BISYNC$V_RCV_COMPLETE,BISYNC$W_STATUS(R4) ; Set that the rcv is cmp
         00E4    30 1F25  5476          BSBW    STOP_RCV_BISYNC                 ; Stop the receiver
  53   00DC D5   0F 1F28  5477          REMQUE  @UCB$Q_XG_RCVS(R5),R3           ; Get an IRP to complete the RCV
             19   1D 1F2D  5478          BVS     10$                             ; If VS then none
     62   42 A2   DE 1F2F  5479          MOVAL   RCV_Z_HEADER(R2),(R2)           ; Set address of the data
        50   59   D0 1F33  5480          MOVL    R9,R0                           ; Set address of buffer
           FD23    30 1F36  5481          BSBW    FINISH_RCV_IO_BISYNC            ; Post Rcv
                     1F39  5482          DSBINT  UCB$B_DIPL(R5)
           F753    30 1F40  5483          BSBW    START_RECEIVE_BISYNC            ; Start next receive
                     1F43  5484          ENBINT
             09   11 1F46  5485          BRB     20$
```

```
                OC A2    59    B0  1F48  5486 10$:     MOVW    R9,RCV_W_MSGSIZ(R2)      ; Set msg size
              00D8 D5    62    0E  1F4C  5487          INSQUE  (R2),@UCB$Q_XG_ATTN+4(R5) ; Insert on waiting for IO list
                                   1F51  5488
                                   1F51  5489 ; If the device is running in half duplex mode, check to see if a transmit can
                                   1F51  5490 ; be send after we receive a message.
                                   1F51  5491
            0A 44 A5    02    E1  1F51  5492 20$:     BBC     #XM$V_CHR_HDPLX,UCB$L_DEVDEPEND(R5),30$ ; If BC not in half duplex
                                   1F56  5493          CLRBIT  #XG_DS_V_XMTING,UCB$W_DEVSTS(R5) ; Set ok to transmit
                      51    D4  1F5B  5494          CLRL    R1
                    ED49    30  1F5D  5495          BSBW    START_TRANSMIT           ; try to send a message
                      0E    11  1F60  5496 30$:     BRB     TIMER_END
                                   1F62  5497
                                   1F62  5498 TIMER_END_UPD:
              00F2 C5    53    B0  1F62  5499          MOVW    R3,UCB$W_XG_RCV_INIT(R5) ; Update # of chars left to input in rcv bu
              00F0 C5    59    B0  1F67  5500          MOVW    R9,UCB$W_XG_RCV_INPR_INDX(R5) ; Update index of buffer to be given t
                OC A4    57    B0  1F6C  5501          MOVW    R7,BISYNC$W_RCV_INDEX(R4) ; Update index of buffer given to device
                                   1F70  5502
                                   1F70  5503 TIMER_END:
                  03FC 8F    BA  1F70  5504          POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9>
                                05  1F74  5505          RSB
                                   1F75  5506
                                   1F75  5507 TIMER_END_CHKCMP:
            F6 0E A4    00    E0  1F75  5508          BBS     #BISYNC$V_RCV_COMPLETE,BISYNC$W_STATUS(R4),TIMER_END ; If BS ignore
                                   1F7A  5509 TIMER_END_CHKCMP_ALT:
                    008F    30  1F7A  5510          BSBW    STOP_RCV_BISYNC          ; Stop receiver
                      52    D5  1F7D  5511          TSTL    R2
                      08    13  1F7F  5512          BEQL    10$
              00E8 D5    62    0E  1F81  5513          INSQUE  (R2),@UCB$Q_XG_FREE+4(R5) ; Return buffer to free list
                    11 A4    B6  1F86  5514          INCW    BISYNC$W_DROP_RCV(R4)    ; Incr the number of droppped msgs
                                   1F89  5515 10$:     DSBINT  UCB$B_DIPL(R5)
                    F703    30  1F90  5516          BSBW    START_RECEIVE_BISYNC     ; Start the receiver
                                   1F93  5517          ENBINT
                                   1F96  5518 ; If the device is running in half duplex mode, check to see if a transmit can
                                   1F96  5519 ; be send after we receive a message.
                                   1F96  5520
            0A 44 A5    02    E1  1F96  5521          BBC     #XM$V_CHR_HDPLX,UCB$L_DEVDEPEND(R5),20$ ; If BC not in half duplex
                                   1F9B  5522          CLRBIT  #XG_DS_V_XMTING,UCB$W_DEVSTS(R5) ; Set ok to transmit
                      51    D4  1FA0  5523          CLRL    R1
                    ED04    30  1FA2  5524          BSBW    START_TRANSMIT           ; try to send a message
                      C9    11  1FA5  5525 20$:     BRB     TIMER_END
                                   1FA7  5526
                                   1FA7  5527 CHECK_XMT:
                2E 0120 C5    E8  1FA7  5528          BLBS    UCB$B_XG_XSTATE(R5),10$ ; IF LBS then XMTer is running
            29 68 A5    0E    E0  1FAC  5529          BBS     #XG_DS_V_CTSTQE_RUN,UCB$W_DEVSTS(R5),10$ ; If BS then CTS TQE runnin
                                   1FB1  5530          DSBINT  UCB$B_DIPL(R5)           ; Sync to device IPL to test for CTS
                51    24 A5    D0  1FB8  5531          MOVL    UCB$L_CRB(R5),R1         ; Get CRB ADDRESS
                51    2C B1    D0  1FBC  5532          MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R1 ; Get CSR address
                04 A1    04    B0  1FC0  5533          MOVW    #XG$C_MODEM,XG$C_MISC_REG(R1) ; Set to get modem IR
                06 A1    10    B3  1FC4  5534          BITW    #XG$M_CTS,XG$C_IND_ADDR(R1) ; If EQL then CTS is high
                                   1FC8  5535          ENBINT
                      10    13  1FCB  5536          BEQL    20$
              0120 C5    01    90  1FCD  5537          MOVB    #XG$C_XMTING,UCB$B_XG_XSTATE(R5) ; Set that device is XMTing
                      54    DD  1FD2  5538          PUSHL   R4
                    ECD2    30  1FD4  5539          BSBW    START_TRANSMIT           ; Start up the transmit
                      54 8ED0  1FD7  5540          POPL    R4
                    FE50    31  1FDA  5541 10$:     BRW     CHECK_RCV                ; Check for receive chars
                  0121 C5    97  1FDD  5542 20$:     DECB    UCB$B_XG_WFCTS_SEC(R5)   ; if GTR then there is time
```

```
                F7   14  1FE1  5543          BGTR    10$                             ; to wait for CTS
                             1FE3  5544 ;
                             1FE3  5545 ; Else set that a modem dsiconnect occured and then abort the transmits
                             1FE3  5546 ;
                             1FE3  5547          SETBIT  #XM$V_STS_DISC,UCB$L_DEVDEPEND(R5)
      51    30 B4    0F      1FE8  5548 30$:     REMQUE  @BISYNC$Q_XMTQ(R4),R1           ; Get next xmt off queue
              13   1D         1FEC  5549          BVS     40$                             ; If VS then no more
      13    0A A1    91      1FEE  5550          CMPB    XMTQ$B_BUFTYP(R1),S^#DYN$C_BUFIO
              14   12         1FF2  5551          BNEQ    50$
      53    0C A1    D0      1FF4  5552          MOVL    XMTQ$L_IRP(R1),R3               ; get the IRP address
      38 A3    2C    3C      1FF8  5553          MOVZWL  #SS$_ABORT,IRP$L_MEDIA(R3)      ; Set abort status
                FC9A   30      1FFC  5554          BSBW    TRANSMIT_IO_DONE
                  E7   11      1FFF  5555          BRB     30$                             ; Get next transmit IRP
           0120 C5     94      2001  5556 40$:     CLRB    UCB$B_XG_XSTATE(R5)
              FE25     31      2005  5557          BRW     CHECK_RCV                       ; Branch to check receive
                             2008  5558
                             2008  5559 50$:     BUG_CHECK       NOBUFPCKT,FATAL
                             200C  5560
                             200C  5561          .IF     DF BISYNC$$$
                             200C  5562 ;
                             200C  5563 ; The following is a test framing routine to be included when using the
                             200C  5564 ; driver to test bisync mode in a non-bisync environment.
                             200C  5565 ;
                             200C  5566 bisync_framing_routine:
                             200C  5567          pushl   r2
                             200C  5568          movl    r0,r2
                             200C  5569          clrl    r0
                             200C  5570          incl    (r2)
                             200C  5571          cmpl    #20,(r2)
                             200C  5572          bgtr    10$
                             200C  5573          setbit  #2,r0
                             200C  5574 10$:     popl    r2
                             200C  5575          rsb
                             200C  5576          .ENDC   ;DF BISYNC$$$
                             200C  5577
```

```
                     200C  5579          .SBTTL  STOP_RECEIVE_BISYNC - Stop the BISYNC mode receiver
                     200C  5580 ;++
                     200C  5581 ; STOP_RECEIVE_BISYNC - Stop the BISYNC mode receiver
                     200C  5582 ;
                     200C  5583 ; This routine stops the current RCV in progress and resets the state of
                     200C  5584 ; the receiver.
                     200C  5585 ;
                     200C  5586 ;       INPUTS
                     200C  5587 ;               R4 = Address of protocol buffer
                     200C  5588 ;               R5 = Address of the UCB
                     200C  5589 ;
                     200C  5590 ;       OUTPUTS
                     200C  5591 ;               R4,R5 are preserved
                     200C  5592 ;               R2 =  Address of buffer to post
                     200C  5593 ;
                     200C  5594 ;--
                     200C  5595 STOP_RCV_BISYNC:
      14 A4    7D    200C  5596          MOVQ    BISYNC$Q_INIT_STATE_INFO(R4),-  ; Reset state information
 015C C5               200F  5597                  UCB$Q_XG_STATE_INFO(R5)
51    24 A5    D0    2012  5598          MOVL    UCB$L_CRB(R5),R1                ; Get CRB address
51    2C B1    D0    2016  5599          MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R1    ; Get CSR address
                     201A  5600          DSBINT  UCB$B_DIPL(R5)                  ; Sync to device IPL
   61    01    AA    2021  5601          BICW2   #XG$M_ENABLE,XG$C_RCV_CSR(R1)   ; Disable the receiver
                     2024  5602          ENBINT                                  ; Lower IPL
                     2027  5603          CLRBIT  #XG_DS_V_RCV_DONEP,UCB$W_DEVSTS(R5) ; Clear that the rcver is busy
                     202C  5604          SETBIT  #BISYNC$V_RCV_COMPLETE,BISYNC$W_STATUS(R4) ; Set that the rcv is com
52 00EC C5    D0    2031  5605          MOVL    UCB$L_XG_RCV_INPR(R5),R2        ; Get receive buffer address
   00EC C5    D4    2036  5606          CLRL    UCB$L_XG_RCV_INPR(R5)           ; Clear buffer address from inpr
   00F0 C5    B4    203A  5607          CLRW    UCB$W_XG_RCV_INPR_INDX(R5)      ; Clear the indices
      0C A4   B4    203E  5608          CLRW    BISYNC$W_RCV_INDEX(R4)
                     05    2041  5609          RSB
                           2042  5610
```

```
                                2042  5612          .SBTTL  REGDUMP - Error log and diagnostics register dump
                                2042  5613 ;++
                                2042  5614 ; REGDUMP -Diagnostics register dump routine
                                2042  5615 ;
                                2042  5616 ; Functional description:
                                2042  5617 ;
                                2042  5618 ; This routine is used to return the DMF-32 diagnostics buffer on error
                                2042  5619 ; or diagnostic QIO function.
                                2042  5620 ;
                                2042  5621 ; INPUTS:
                                2042  5622 ;
                                2042  5623 ;       R0 = ADDRESS OF THE BUFFER
                                2042  5624 ;       R5 = UCB ADDRESS OF THE UNIT
                                2042  5625 ;
                                2042  5626 ; OUTPUTS:
                                2042  5627 ;
                                2042  5628 ;       R0,R1 ARE USED
                                2042  5629 ;       R5 = UCB ADDRESS OF THE UNIT
                                2042  5630 ;--
                                2042  5631 REGDUMP:
                 80    03   9A   2042  5632          MOVZBL  #3,(R0)+                ; Insert number of returned long words
                       0B   E1   2045  5633          BBC     #XM$V_STS_ACTIVE,-      ; BR if not active
                    0C 44 A5     2047  5634                  UCB$L_DEVDEPEND(R5),20$
                                 204A  5635
                                 204A  5636          ASSUME  UCB$W_XG_XMTCSR EQ UCB$W_XG_RCVCSR+2
                                 204A  5637          ASSUME  UCB$W_XG_RCVERR EQ UCB$W_XG_XMTCSR+2
                                 204A  5638          ASSUME  UCB$W_XG_XMTERR EQ UCB$W_XG_RCVERR+2
                                 204A  5639          ASSUME  UCB$W_XG_DSC EQ UCB$W_XG_XMTERR+2
                                 204A  5640
              80    0122 C5    7D   204A  5641          MOVQ    UCB$W_XG_RCVCSR(R5),(R0)+
              80    012A C5    3C   204F  5642          MOVZWL  UCB$W_XG_DSC(R5),(R0)+
                          04    11   2054  5643          BRB     30$                     ; Continue
                          80    7C   2056  5644 20$:     CLRQ    (R0)+                   ; Return no errors
                          60    D4   2058  5645          CLRL    (R0)                    ;
                                05   205A  5646 30$:     RSB
                                     205B  5647
```

```
                                205B  5649          .SBTTL  Poke user process on attention condition
                                205B  5650
                                205B  5651 ;++
                                205B  5652 ; POKE_USER - Poke user process on attention condition
                                205B  5653 ;
                                205B  5654 ; FUNCTIONAL DESCRIPTION:
                                205B  5655 ;
                                205B  5656 ; This routine is used when data is avaliable or the unit is shutdown.
                                205B  5657 ; The action is to declare the ast's and send a message to the assoc. mailbox.
                                205B  5658 ;
                                205B  5659 ; INPUTS:
                                205B  5660 ;
                                205B  5661 ;       R4 = Message type -- 0 if no message
                                205B  5662 ;       R5 = Unit UCB address
                                205B  5663 ;
                                205B  5664 ; OUTPUTS:
                                205B  5665 ;
                                205B  5666 ;       R0 = Low bit clear only if user is not notified
                                205B  5667 ;       R5 = UCB ADDRESS
                                205B  5668 ;--
                                205B  5669 POKE_USER:                                           ; POKE USER
                                205B  5670          DSBINT  UCB$B_FIPL(R5)                      ; Sync to Fork
                          7E    D4  2062  5671          CLRL    -(SP)                           ; Assume failure
                51  0090    C5    9E  2064  5672          MOVAB   UCB$L_XG_AST(R5),R1             ; Get AST listhead
                          61    D5  2069  5673          TSTL    (R1)                            ; Empty ?
                          17    13  206B  5674          BEQL    17$                             ; If so, branch
                          6E    D6  206D  5675          INCL    (SP)                            ; Indicate success
                      54  51    D0  206F  5676          MOVL    R1,R4                           ; Copy list head address
                      51  61    D0  2072  5677 10$:     MOVL    (R1),R1                         ; Address a block
                          07    13  2075  5678          BEQL    15$                             ; If EQL then done
              1C A1    44  A5    D0  2077  5679          MOVL    UCB$L_DEVDEPEND(R5),^X01C(R1)   ; Change param
                          F4    11  207C  5680          BRB     10$
                00000000'GF    16  207E  5681 15$:     JSB     G^COM$DELATTNAST                ; Deliver AST'S
                          50  8ED0  2084  5682 17$:     POPL    R0                              ; Set status
                                    2087  5683          ENBINT                                  ; Enable interrupts
                              05    208A  5684          RSB
```

```
                                208B  5686          .SBTTL  TIMEOUT - TIMEOUT
                                208B  5687 ;++
                                208B  5688 ; TIMEOUT - Timeout
                                208B  5689 ;
                                208B  5690 ; FUNCTIONAL DESCRIPTION:
                                208B  5691 ;
                                208B  5692 ; This routine is entered on device timeout. The action is to shut
                                208B  5693 ; the unit down.
                                208B  5694 ;
                                208B  5695 ; INPUTS:
                                208B  5696 ;
                                208B  5697 ;       R5 = UCB address
                                208B  5698 ;
                                208B  5699 ; OUTPUTS:
                                208B  5700 ;
                                208B  5701 ;       NONE
                                208B  5702 ;--
                         0002'  208B  5703          .WORD   TIMEOUT-.
                                208D  5704 TIMEOUT:
                    0B    E1    208D  5705          BBC     #XM$V_STS_ACTIVE,-      ; Br BC device is not active
                12 44 A5        208F  5706                   UCB$L_DEVDEPEND(R5),20$
               54  022C 8F    3C 2092  5707          MOVZWL  #SS$_TIMEOUT,R4         ; Assume timeout
                    05    E1    2097  5708 10$:     BBC     #UCB$V_POWER,-          ; Br unless powerfail
                08 64 A5        2099  5709                   UCB$W_STS(R5),20$
               54  0364 8F    3C 209C  5710          MOVZWL  #SS$_POWERFAIL,R4       ; Indicate powerfail
                    F966    30 20A1  5711          BSBW    SCHED_FORK              ; Create the error fork process
                    05      20A4  5712 20$:     RSB
                            20A5  5713
```

```
                                20A5  5715          .SBTTL  STOP_BISYNC_TIMER - Stop the bisync mode timer
                                20A5  5716 ;++
                                20A5  5717 ; STOP_BISYNC_TIMER - Stop the bisync mode timer
                                20A5  5718 ;
                                20A5  5719 ; This routine is called to stop the timer from running.
                                20A5  5720 ;
                                20A5  5721 ;       INPUTS
                                20A5  5722 ;               R5 = UCB address
                                20A5  5723 ;
                                20A5  5724 ;       OUTPUTS
                                20A5  5725 ;               The BISYNC mode timer is taken off the TQE queue
                                20A5  5726 ;
                                20A5  5727 ;--
                                20A5  5728 STOP_BISYNC_TIMER:
          54    00D0 C5   D0    20A5  5729          MOVL    UCB$A_XG_PRO_BUFFER(R5),R4      ; Get protocol buffer address
                     30   BB    20AA  5730          PUSHR   #^M<R4,R5>
            50    38 A4   DE    20AC  5731          MOVAL   BISYNC$L_TQE(R4),R0             ; Get protocol TQE address
            55    14 A0   D0    20B0  5732          MOVL    TQE$L_FR4(R0),R5                ; Get address of protocol buffer
                                20B4  5733
                                20B4  5734 ; Set in FPC to ensure proper deletion of the TQE, that is not more than
                                20B4  5735 ; one is deleted. If just the FPC is check then all TQEs' using this code
                                20B4  5736 ; (any device running BISYNC) would be affected.
                                20B4  5737
           0C A0    55   D0    20B4  5738          MOVL    R5,TQE$L_FPC(R0)
              54    01   9A    20B8  5739          MOVZBL  #TQE$C_SSSNGL,R4                ; SET system proc not repeatable
          00000000'GF   16    20BB  5740          JSB     G^EXE$RMVTIMQ                   ; Call system routine to remove TQE
                     30   BA    20C1  5741          POPR    #^M<R4,R5>
                                20C3  5742          CLRBIT  #BISYNC$V_TIMER_RUNNING,BISYNC$W_STATUS(R4) ; Set timer not running
     0000000D'EF    04   8A    20C8  5743          BICB    #DPT$M_NOUNLOAD,DPT$TAB+DPT$B_FLAGS ; Set driver reloadable
                           05   20CF  5744          RSB
```

```
                                20D0  5746          .SBTTL  CANCEL - Cancel I/O rotuine
                                20D0  5747 ;++
                                20D0  5748 ;CANCEL - Cancels all I/O in progress
                                20D0  5749 ;
                                20D0  5750 ; By looking at the channel number saved at line startup time, cancel
                                20D0  5751 ; determines which entity (line or circuit) to shut down.
                                20D0  5752 ;
                                20D0  5753 ;       INPUTS  R2 = channel number
                                20D0  5754 ;               R3 = current IRP address
                                20D0  5755 ;               R4 = PCB address
                                20D0  5756 ;               R5 = UCB address
                                20D0  5757 ;               R8 = Cancel reason code (zero vanilla flavored cancel)
                                20D0  5758 ;
                                20D0  5759 ;               IPL = FIPL
                                20D0  5760 ;
                                20D0  5761 ;       OUTPUTS R0 - R3 are destroyed
                                20D0  5762 ;
                                20D0  5763 ;--
                                20D0  5764 CANCEL:
                                20D0  5765
                                20D0  5766 ; On  CANCEL or DEASSIGN of the channel check to see if the line should
                                20D0  5767 ; drop DTR. If this is the last DEASSIGN then the line must drop DTR and
                                20D0  5768 ; we should reinitialize the settable parameters to their default values.
                                20D0  5769
                      51   D4   20D0  5770          CLRL    R1                              ; Assume DTR can be dropped
                    5C A5  B5   20D2  5771          TSTW    UCB$W_REFC(R5)                  ; If reference count is zero
                      07   13   20D5  5772          BEQL    10$                             ; this is last the DEASSIGN
              51  0156 C5   9A   20D7  5773          MOVZBL  UCB$B_XG_MODEM_CLR(R5),R1       ; Set whether or not DTR
                                20DC  5774                                                  ; should be dropped
                      07   11   20DC  5775          BRB     20$                             ; Do not reinit param values
                                20DE  5776
                      06   BB   20DE  5777 10$:     PUSHR   #^M<R1,R2>
                    E106   30   20E0  5778          BSBW    INIT_PARAM                      ; Init params to known values
                      06   BA   20E3  5779          POPR    #^M<R1,R2>
                                20E5  5780
              50  0167 C5   D0   20E5  5781 20$:     MOVL    UCB$B_XG_PROTYPE(R5),R0         ; Get protocol
                                20EA  5782          CASE    R0,TYPE=B,<-
                                20EA  5783                  30$,-                           ; Lapb
                                20EA  5784                  30$,-                           ; Bisync
                                20EA  5785                  >                               ; Fall thru on DDCMP
              52  0130 C5   B1   20F2  5786          CMPW    UCB$W_XG_CHANL(R5),R2           ; If EQL then matches channel
                      07   13   20F7  5787          BEQL    SHUTDOWN_LINE_ALT               ; assigned to line
                    00DA   31   20F9  5788          BRW     SHUTDOWN_CIRCUIT                ; Else shutdown the circuit
                                20FC  5789
                      02   11   20FC  5790 30$:     BRB     SHUTDOWN_LINE_ALT
                                20FE  5791
```

```
                     20FE  5793
                     20FE  5794          .SBTTL SHUTDOWN - Shut down unit, device and/or protocol
                     20FE  5795 ;++
                     20FE  5796 ; SHUTDOWN_LINE - Shut down unit
                     20FE  5797 ;
                     20FE  5798 ; FUNCTIONAL DESCRIPTION:
                     20FE  5799 ;
                     20FE  5800 ; This routine is used to shut down the unit as a result of a
                     20FE  5801 ; setmode shutdown on the controller or a fatal error. The mapping
                     20FE  5802 ; registers are returned, a call is made to shutdown the circuit; and
                     20FE  5803 ; all quotas are returned.
                     20FE  5804 ;
                     20FE  5805 ; INPUTS:
                     20FE  5806 ;
                     20FE  5807 ;       R5 = UCB address
                     20FE  5808 ;
                     20FE  5809 ; For SHUTDOWN_LINE_ALT
                     20FE  5810 ;
                     20FE  5811 ;       R1 = 0 if DTR can be dropped on shut down
                     20FE  5812 ;            1 if DTR can not be dropped on shut down
                     20FE  5813 ;
                     20FE  5814 ; OUTPUTS:
                     20FE  5815 ;
                     20FE  5816 ;       R5 = UCB address
                     20FE  5817 ;       R6 - R9 are preserved
                     20FE  5818 ;
                     20FE  5819 ;       R0-R3 are destroyed.
                     20FE  5820 ;--
                     20FE  5821 SHUTDOWN_LINE:                                   ; Shut down unit
           51   D4  20FE  5822          CLRL    R1
                     2100  5823 SHUTDOWN_LINE_ALT:
           02   E1  2100  5824          BBC     #XG_DS_V_INITED,-
        05 68 A5       2102  5825                  UCB$W_DEVSTS(R5),5$             ; BR if not inited
           04   E0  2105  5826          BBS     #UCB$V_ONLINE,-
        06 64 A5       2107  5827                  UCB$W_STS(R5),10$               ; Br if online
     50   20D4 8F   3C  210A  5828 5$:    MOVZWL  #SS$_DEVINACT,R0                ; Set status
                05   210F  5829          RSB                                     ; ... and return
                     2110  5830
         03DC 8F   BB  2110  5831 10$:    PUSHR   #^M<R2,R3,R4,R6,R7,R8,R9>       ; Save the registers
                     2114  5832
      64 A5    23   AA  2114  5833          BICW    #UCB$M_INT!UCB$M_POWER!-
                     2118  5834                  UCB$M_TIM,UCB$W_STS(R5)         ; Reset device status
                AA   2118  5835          BICW    #^C<XG_DS_M_XMTING!-            ; Clear all but xmt off
                     2119  5836                      XG_DS_M_RCVING!-            ; rcv off
                     2119  5837                      XG_DS_M_FORK_PEND!-         ; and fork pending bits
                     2119  5838                      XG_DS_M_CTSTQE_RUN!-        ; CTSTQE running
                     2119  5839                      XG_DS_M_CLEAN>,-            ; Clean IRP outstanding
68 A5    3FF4 8F        2119  5840                  UCB$W_DEVSTS(R5)
                     211E  5841
                     211E  5842 ;
                     211E  5843 ; Clear the modem register on the shutdown. Includes dropping DTR.
                     211E  5844 ;
                     211E  5845
         1A 51    E8  211E  5846          BLBS    R1,15$                          ; If BS then don't drop DTR
      54   24 A5    D0  2121  5847          MOVL    UCB$L_CRB(R5),R4                ; Get unit csr
      54   2C B4    D0  2125  5848          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4
                     2129  5849          DSBINT  UCB$B_DIPL(R5)                  ; Disable device interrupts
```

```
                04 A4    04    B0  2130  5850          MOVW    #XG$C_MODEM,XG$C_MISC_REG(R4)   ; Get the modem indirect reg
                06 A4    00    B0  2134  5851          MOVW    #0,XG$C_IND_ADDR(R4)            ; Clear modem flags
                                   2138  5852          ENBINT                                  ; Return to FIPL
                      0098    30   213B  5853 15$:     BSBW    SHUTDOWN_CIRCUIT                ; Shutdown the circuit
              50    0167 C5    9A   213E  5854          MOVZBL  UCB$B_XG_PROTYPE(R5),R0         ; Get protocol type
                                   2143  5855          CASE    R0,TYPE=B,<-                    ; Case on protocol type
                                   2143  5856                  18$,-                           ; LAPB
                                   2143  5857                  16$,-                           ; BISYNC
                                   2143  5858                  >                               ; Fall thru on DDCMP
                                   214B  5859
                                   214B  5860 ;
                                   214B  5861 ; Stop the DDCMP timer. First set up registers with params for call
                                   214B  5862 ; then make the call to the protocol.
                                   214B  5863 ;
                      55    DD   214B  5864          PUSHL   R5
                   56   07    9A   214D  5865          MOVZBL  #DLK$C_STOP_TIMER,R6            ; Set up R6 with the DDCMP comm
                      57    7C   2150  5866          CLRQ    R7                              ; CLR all registers
                      59    D4   2152  5867          CLRL    R9                              ; Clear R9
              55    00D0 C5    D0   2154  5868          MOVL    UCB$A_XG_PRO_BUFFER(R5),R5      ; Set up R5 with TF block addr
                 DEA4'    30   2159  5869          BSBW    DDCMP                           ; Call protocol
                      55 8ED0   215C  5870          POPL    R5
                                   215F  5871
                                   215F  5872 ; After the timer has been stopped it is ok to reload the driver.
                                   215F  5873 ;
        0000000D'EF    04    8A   215F  5874          BICB    #DPT$M_NOUNLOAD,DPT$TAB+DPT$B_FLAGS
                      03    11   2166  5875          BRB     18$
                                   2168  5876
                   FF3A    30   2168  5877 16$:     BSBW    STOP_BISYNC_TIMER               ; Stop the BISYNC timer
                                   216B  5878
                                   216B  5879 ;
                                   216B  5880 ; Release the map registers
                                   216B  5881 ;
                54   24 A5    D0   216B  5882 18$:     MOVL    UCB$L_CRB(R5),R4                ; Address CRB
                   57    04    9A   216F  5883          MOVZBL  #NUM_MAP_REG,R7                 ; Set number of rcv blocks
              56   0110 C5    9E   2172  5884          MOVAB   UCB$Z_XG_VECTOR(R5),R6          ; Address rcv block vector
                      86    D0   2177  5885 20$:     MOVL    (R6)+,-                         ; Setup for call
                    34 A4         2179  5886                  CRB$L_INTD+VEC$W_MAPREG(R4)
                      0A    19   217B  5887          BLSS    30$                             ; If LSS then done
                                   217D  5888          RELMPR                                  ; Give map regs to EXEC
                FC A6    01    CE   2183  5889          MNEGL   #1,-4(R6)                       ; Indicate done
                   ED 57    F5   2187  5890 30$:     SOBGTR  R7,20$                          ; If EQL then done
                                   218A  5891 ;
                                   218A  5892 ; Deallocate the protocol buffer
                                   218A  5893 ;
              50    00D0 C5    D0   218A  5894          MOVL    UCB$A_XG_PRO_BUFFER(R5),R0      ; Get the address of the buffer
                   00D0 C5    D4   218F  5895          CLRL    UCB$A_XG_PRO_BUFFER(R5)         ; Clear knowledge of the buffer
                52   08 A0    3C   2193  5896          MOVZWL  UCB$W_SIZE(R0),R2               ; Get the size
                      24    BB   2197  5897          PUSHR   #^M<R2,R5>
               00000000'GF    16   2199  5898          JSB     G^COM$DRVDEALMEM                ; Dealloc the buffer
                      24    BA   219F  5899          POPR    #^M<R2,R5>
                                   21A1  5900
                                   21A1  5901 ;
                                   21A1  5902 ; Restore the buffered I/O quota and protocol buffer quota to the starter
                                   21A1  5903 ;
              50    012C C5    3C   21A1  5904          MOVZWL  UCB$L_XG_PID(R5),R0             ; Get pid of last starter
        51   00000000'GF    D0   21A6  5905          MOVL    G^SCH$GL_PCBVEC,R1              ; Address PCB vector
                50   6140    D0   21AD  5906          MOVL    (R1)[R0],R0                     ; Get PCB of owner
```

```
                60 A0   D1  21B1  5907          CMPL    PCB$L_PID(R0),-
            012C C5         21B4  5908                  UCB$L_XG_PID(R5)            ; Still there?
                 15   12    21B7  5909          BNEQ    40$                         ; If NOT then BR
     50     0080 C0   D0    21B9  5910          MOVL    PCB$L_JIB(R0),R0            ; Get JIB address
     51     010C C5   3C    21BE  5911          MOVZWL  UCB$W_XG_QUOTA(R5),R1       ; Convert to longword
            52   51   C0    21C3  5912          ADDL    R1,R2                       ; Get bytes quota to return
       20 A0   52   C0    21C6  5913          ADDL    R2,JIB$L_BYTCNT(R0)         ; Return byte count quota
            010C C5   B4    21CA  5914          CLRW    UCB$W_XG_QUOTA(R5)          ; Prevent this from being
                      21CE  5915                                                  ; returned again
            50   01   3C    21CE  5916 40$:     MOVZWL  S^#SS$_NORMAL,R0            ; Set status
            03DC 8F   BA    21D1  5917          POPR    #^M<R2,R3,R4,R6,R7,R8,R9>   ; Save the registers
                      05    21D5  5918          RSB
                            21D6  5919
```

```
                        21D6  5921
                        21D6  5922 ;++
                        21D6  5923 ; SHUTDOWN_CIRCUIT - Shut down device and protocol
                        21D6  5924 ;
                        21D6  5925 ; FUNCTIONAL DESCRIPTION:
                        21D6  5926 ;
                        21D6  5927 ; This routine is used to shut down the circuit as a result of a setmode
                        21D6  5928 ; shutdown on the tributary or on the controller or by a fatal error on
                        21D6  5929 ; the device. The routines frees allocated blocks; completes IRP's with
                        21D6  5930 ; the active bit clear; and halts the protocol.
                        21D6  5931 ;
                        21D6  5932 ; INPUTS:
                        21D6  5933 ;
                        21D6  5934 ;       R5 = UCB address
                        21D6  5935 ;
                        21D6  5936 ; OUTPUTS:
                        21D6  5937 ;
                        21D6  5938 ;       R5 = UCB address
                        21D6  5939 ;       R6 - R9 are preserved
                        21D6  5940 ;
                        21D6  5941 ;       R0-R3 are destroyed.
                        21D6  5942 ;--
                        21D6  5943 SHUTDOWN_CIRCUIT:
          0B   E0       21D6  5944         BBS     #XM$V_STS_ACTIVE,-                  ; If BS then trib is active
      06 44 A5                 21D8  5945                 UCB$L_DEVDEPEND(R5),5$              ; Else return with
50    20D4 8F   3C       21DB  5946         MOVZWL  #SS$_DEVINACT,R0                    ; status set
                05       21E0  5947         RSB
                         21E1  5948
    03DC 8F     BB       21E1  5949 5$:     PUSHR   #^M<R2,R3,R4,R6,R7,R8,R9>           ; Save the registers
54    24 A5     D0       21E5  5950         MOVL    UCB$L_CRB(R5),R4                    ; Get unit csr
54    2C B4     D0       21E9  5951         MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4
                         21ED  5952         DSBINT  UCB$B_DIPL(R5)                      ; Raise IPL for master clear
    0800 8F     AA       21F4  5953         BICW    #XM$M_STS_ACTIVE,-
      44 A5              21F8  5954                 UCB$L_DEVDEPEND(R5)                 ; Set inactive
                A8       21FA  5955         BISW    #<XG_DS_M_XMTING!-                  ; Set the XMTer and
                         21FB  5956                   XG_DS_M_RCVING>,-                 ; RCVer off
68 A5   03               21FB  5957                 UCB$W_DEVSTS(R5)
                         21FE  5958
                         21FE  5959         ASSUME XG$C_IDLE EQ 0
                         21FE  5960
    0120 C5     94       21FE  5961         CLRB    UCB$B_XG_XSTATE(R5)                 ; Reset xmter state
                         2202  5962
                         2202  5963         TIMEWAIT #50,#XG$M_ENABLE,XG$C_XMT_CSR(R4),W,EQL ; Wait till XMT'r
                         2226  5964                                                     ; completes
                         2226  5965
    0080 8F     A8       2226  5966         BISW    #XG$M_MASTER_RESET,-                ; Master clear the unit
      04 A4              222A  5967                 XG$C_MISC_REG(R4)
                         222C  5968
    0104 C5     D4       222C  5969         CLRL    UCB$L_XG_XMTEND(R5)                 ; Clear XMT timeout
          02    E1       2230  5970         BBC     #XG_DS_V_INITED,-
   32 68 A5              2232  5971                 UCB$W_DEVSTS(R5),10$                ; BR if not inited
          1F    BB       2235  5972         PUSHR   #^M<R0,R1,R2,R3,R4>                 ; Save registers
54    24 A5     D0       2237  5973         MOVL    UCB$L_CRB(R5),R4                    ; Get CRB address
54    38 B4     D0       223B  5974         MOVL    @CRB$L_INTD+VEC$L_ADP(R4),R4        ; Get addr of configuration reg
   50   04     D0       223F  5975         MOVL    #NUM_MAP_REG,R0                     ; Set number of mapping regs
52  0110 C5     DE       2242  5976         MOVAL   UCB$Z_XG_VECTOR(R5),R2              ; Get mapping slot
          00    EF       2247  5977 6$:     EXTZV   #VEC$V_MAPREG,-                     ; Get start of mapping regs
```

```
                51   62   0F        2249  5978                    #VEC$S_MAPREG,(R2),R1
                                    224C  5979
                                    224C  5980          ASSUME    VEC$B_NUMREG EQ VEC$W_MAPREG+2
                                    224C  5981
                  53   02 A2   9A   224C  5982          MOVZBL    2(R2),R3                        ; Get number of map regs
              51    0800 C441   DE   2250  5983          MOVAL     UBA$L_MAP(R4)[R1],R1            ; Get first register
            81   80000000 8F   CA   2256  5984 8$:       BICL      #UBA$M_MAP_VALID,(R1)+          ; Mark the register invalid
                       F6 53   F5   225D  5985          SOBGTR    R3,8$                           ; Loop thru all registers
                          82   D5   2260  5986          TSTL      (R2)+                           ; Get next slot
                       E2 50   F5   2262  5987          SOBGTR    R0,6$                           ; Loop thru all slots
                          1F   BA   2265  5988          POPR      #^M<R0,R1,R2,R3,R4>             ; Restore registers
                                    2267  5989
                                    2267  5990 ;
                                    2267  5991 ; Clear any RCV's or XMT's pending
                                    2267  5992 ;
                                    2267  5993 ; Special case the BISYNC case, because we do not use the inpr queu as
                                    2267  5994 ; a queue. For Bisync, we can have only one receive outstanding at a time
                                    2267  5995 ; thus there is no need to use the queue and in fact the second long
                                    2267  5996 ; word is used for other information necessary to receive data.
                                    2267  5997 ;
            0167 C5   01   91       2267  5998 10$:       CMPB      #XG$C_PROTYPE_BISYNC,UCB$B_XG_PROTYPE(R5) ; Bisync?
                          0E   13   226C  5999          BEQL      11$                             ; If eql then yes
            56    00EC D5   0F      226E  6000          REMQUE    @UCB$Q_XG_RCV_INPR(R5),R6       ; Anything on the inpr queues
                          27   1D   2273  6001          BVS       13$                             ; If VS then no
            00E8 D5       66   0E   2275  6002          INSQUE    (R6),@UCB$Q_XG_FREE+4(R5)       ; Else queue to free queue
                          EB   11   227A  6003          BRB       10$
            56    00EC C5   00      227C  6004 11$:       MOVL      UCB$L_XG_RCV_INPR(R5),R6        ; Get the rcv buffer
                          09   13   2281  6005          BEQL      12$                             ; No buffer
                 00EC C5   D4      2283  6006          CLRL      UCB$L_XG_RCV_INPR(R5)           ; Clear the address from slot
            00E8 D5       66   0E   2287  6007          INSQUE    (R6),@UCB$Q_XG_FREE+4(R5)       ; Insert on free queue
            54    00D0 C5   00      228C  6008 12$:       MOVL      UCB$A_XG_PRO_BUFFER(R5),R4      ; Get protocol buffer address
              56    1C A4   00      2291  6009          MOVL      BISYNC$A_RCV_BUFFER(R4),R6      ; Get addr of buff given to dev
                          05   13   2295  6010          BEQL      13$                             ; No buffer
            00E8 D5       66   0E   2297  6011          INSQUE    (R6),@UCB$Q_XG_FREE+4(R5)       ; Insert on free queue
                          53   D4   229C  6012 13$:       CLRL      R3                              ; Get prim buffer to compl
          51    00FC C543   D0      229E  6013          MOVL      UCB$Z_XG_XMT_INPR(R5)[R3],R1    ; Comp XMT's on inprogress Q
                          13   13   22A4  6014          BEQL      14$                             ; If EQL none to complete
                00FC C543   D4      22A6  6015          CLRL      UCB$Z_XG_XMT_INPR(R5)[R3]
                  0164 C5   97      22AB  6016          DECB      UCB$B_XG_XMTCNT(R5)             ; Decr # of outstanding XMT's
                          02   E0   22AF  6017          BBS       #XMTQ$V_CONTROL,-               ; If BS then control message
                 05 1F A1           22B1  6018                    XMTQ$B_FLAG(R1),14$             ; don't put on post queue
            00F8 D5       61   0E   22B4  6019          INSQUE    (R1),@UCB$Q_XG_POST+4(R5)       ; Insert to be completed
                                    22B9  6020 14$:       ENBINT                                  ; Return to fork level
                                    22BC  6021
                                    22BC  6022 ;
                                    22BC  6023 ; Deallocate all the attention AST control blocks
                                    22BC  6024 ;
            57    0090 C5   9E      22BC  6025 15$:       MOVAB     UCB$L_XG_AST(R5),R7             ; Address list head for AST's
                  50    67   D0      22C1  6026          MOVL      (R7),R0                         ; Anything in the list?
                          1B   13   22C4  6027          BEQL      20$                             ; If EQL then empty
                    56    22 A0   3C   22C6  6028          MOVZWL    ACB$L_KAST+10(R0),R6            ; Force channel
                    52    24 A0   3C   22CA  6029          MOVZWL    ACB$L_KAST+12(R0),R2            ; Get PID index
          54   00000000'GF       D0   22CE  6030          MOVL      G^SCH$GL_PCBVEC,R4
                    54    6442   D0   22D5  6031          MOVL      (R4)[R2],R4                     ; Get PCB
                 00000000'GF       16   22D9  6032          JSB       G^COM$FLUSHATTNS                ; Flush the attention AST's
                          DB   11   22DF  6033          BRB       15$                             ; Continue until done
                                    22E1  6034 ;
```

```
                                22E1  6035 ; Clear the post queue
                                22E1  6036 ;
                                22E1  6037
                     56  00F4 D5  0F  22E1  6038 20$:    REMQUE  @UCB$Q_XG_POST(R5),R6           ; Get next buffer to complete
                                 25  1D  22E6  6039          BVS     30$                         ; If VS then no buffer found
                       13  0A A6  91  22E8  6040          CMPB    IRP$B_TYPE(R6),S^#DYN$C_BUFIO ; If NEQ then not XMT buffer
                                 12  12  22EC  6041          BNEQ    25$
                       53  0C A6  D0  22EE  6042          MOVL    XMTQ$L_IRP(R6),R3           ; If EQL then no IRP to comp
                                 ED  13  22F2  6043          BEQL    20$
                           51    56  D0  22F4  6044          MOVL    R6,R1                       ; Set up R1 for branch
                     38 A3    2C  3C  22F7  6045          MOVZWL  #SS$_ABORT,IRP$L_MEDIA(R3)  ; Set abort status and compl
                             F99B  30  22FB  6046          BSBW    TRANSMIT_IO_DONE            ; the request
                                 E1  11  22FE  6047          BRB     20$
                       17  0A A6  91  2300  6048 25$:    CMPB    IRP$B_TYPE(R6),S^#DYN$C_NET ; If EQL then RCV buffer
                                 30  12  2304  6049          BNEQ    55$
                    00E8 D5    66  0E  2306  6050          INSQUE  (R6),@UCB$Q_XG_FREE+4(R5)   ; Put on free Q to complete
                                 D4  11  2308  6051          BRB     20$
                                     230D  6052 ;
                                     230D  6053 ; Deallocate all receive buffers
                                     230D  6054 ;
                     56  00E4 C5  9E  230D  6055 30$:    MOVAB   UCB$Q_XG_FREE(R5),R6        ; Get queue listhead
                                 0D  10  2312  6056          BSBB    40$                         ; Empty queue
                     56  00D4 C5  9E  2314  6057          MOVAB   UCB$Q_XG_ATTN(R5),R6        ; Get queue listhead
                                 06  10  2319  6058          BSBB    40$                         ; Empty queue
                         010E C5  94  231B  6059          CLRB    UCB$B_XG_INUS(R5)           ; Free all slots
                                 19  11  231F  6060          BRB     60$                         ; Continue
                       50  00 B6  0F  2321  6061 40$:    REMQUE  @(R6),R0                    ; Get buffer
                                 0E  1D  2325  6062          BVS     50$                         ; If VS then none
                           42 A5  A0  2327  6063          ADDW    UCB$W_DEVBUFSIZ(R5),-       ; Restore quota
                         010C C5      232A  6064                  UCB$W_XG_QUOTA(R5)
                    00000000'GF  16  232D  6065          JSB     G^COM$DRVDEALMEM            ; Deallocate buffer
                                 EC  11  2333  6066          BRB     40$                         ; Loop
                                 05  2335  6067 50$:    RSB
                                     2336  6068
                                     2336  6069 55$:    BUG_CHECK NOBUFPCKT,FATAL
                                     233A  6070
                                     233A  6071 ;
                                     233A  6072 ; Complete all associated receive IO packets
                                     233A  6073 ;
                     53  00DC D5  0F  233A  6074 60$:    REMQUE  @UCB$Q_XG_RCVS(R5),R3       ; If VS then queue is empty
                                 09  1D  233F  6075          BVS     70$
                       38 A3    2C  3C  2341  6076          MOVZWL  #SS$_ABORT,IRP$L_MEDIA(R3)  ; Set status and size
                             F963  30  2345  6077          BSBW    IO_DONE                     ; Complete the request
                                 F0  11  2348  6078          BRB     60$                         ; Branch to get next IRP
                                     234A  6079 ;
                                     234A  6080 ; Complete all XMITS
                                     234A  6081 ;
                                     234A  6082 70$:    DSBINT  UCB$B_DIPL(R5)              ; Lock out interrupts
                                     2351  6083
                                 03  AA  2351  6084          BICW    #UCB$M_TIM!UCB$M_INT,-      ;   "   "       "
                           64 A5      2353  6085                  UCB$W_STS(R5)
                                     2355  6086          ENBINT                              ; Reset interrupts
                    0A 68 A5    0F  E5  2358  6087          BBCC    #XG_DS_V_CLEAN,UCB$W_DEVSTS(R5),73$ ; IF BS then do clean handling
                                 53  D4  235D  6088          CLRL    R3                          ; Set no buffer to post
                     54  00D0 C5  D0  235F  6089          MOVL    UCB$A_XG_PRO_BUFFER(R5),R4  ; Set protocol buffer address
                             EC33  30  2364  6090          BSBW    CLEAN_FORK_ENTRY
                                 28  BB  2367  6091 73$:    PUSHR   #^M<R3,R5>                  ; Save registers
```

```
                                                E 13
50   0167 C5   9A  2369  6092          MOVZBL  UCB$B_XG_PROTYPE(R5),R0          ; Get protocol type
55   00D0 C5   D0  236E  6093          MOVL    UCB$A_XG_PRO_BUFFER(R5),R5       ; Set protocol buffer address
                   2373  6094          CASE    R0,TYPE=B,<-                     ; Case on protocol type
                   2373  6095                  75$,-                            ; LAPB
                   2373  6096                  75$,-                            ; BISYNC
                   2373  6097                  >                                ; Fall thru on DDCMP
     56   04   9A  237B  6098          MOVZBL  #DLK$C_USRINT,R6                 ; Set up to halt the protocol
     57   02   9A  237E  6099          MOVZBL  #DLK$M_STOP,R7
          58   7C  2381  6100          CLRQ    R8
        DC7A' 30  2383  6101          BSBW    DDCMP
56   28 A5   9E  2386  6102          MOVAB   TF$Q_CMPQ(R5),R6                 ; Get queue to complete from
          28   BA  238A  6103          POPR    #^M<R3,R5>                       ; Restore the registers
          0E   11  238C  6104          BRB     80$
56   20 A5   9E  238E  6105 75$:      MOVAB   LAPB$Q_XMTQ(R5),R6               ; Get start of device queue
          28   BA  2392  6106          POPR    #^M<R3,R5>                       ; Restore the registers
          06   11  2394  6107          BRB     80$
56   30 A5   9E  2396  6108 78$:      MOVAB   BISYNC$Q_XMTQ(R5),R6             ; Get start of device queue
          28   BA  239A  6109          POPR    #^M<R3,R5>                       ; Restore the registers
51   00 B6   0F  239C  6110 80$:      REMQUE  @(R6),R1                         ; Complete all XMT's
          13   1D  23A0  6111          BVS     90$                              ; If VS then no more to cmplt
13   0A A1   91  23A2  6112          CMPB    XMTQ$B_BUFTYP(R1),S^#DYN$C_BUFIO ; Branch NEQ not a valid buff
          15   12  23A6  6113          BNEQ    100$
53   0C A1   D0  23A8  6114          MOVL    XMTQ$L_IRP(R1),R3                ; Get associated IRP
38 A3    2C   3C  23AC  6115          MOVZWL  #SS$_ABORT,IRP$L_MEDIA(R3)       ; Set aborted status
       F8E6   30  23B0  6116          BSBW    TRANSMIT_IO_DONE                 ; Complete the I/O
          E7   11  23B3  6117          BRB     80$                              ; Branch to get next
     50   01   3C  23B5  6118 90$:      MOVZWL  S^#SS$_NORMAL,R0                 ; Assume shutdown
     03DC 8F   BA  23B8  6119          POPR    #^M<R2,R3,R4,R6,R7,R8,R9>        ; Save the registers
          05   23BC  6120          RSB
               23BD  6121
               23BD  6122 100$:     BUG_CHECK       NOBUFPCKT,FATAL
               23C1  6123
```

```
                                        .SBTTL  VALIDATE_P2,  Validate P2 buffer parameters
                                        .SBTTL  VALIDATE_P2_TRIB,  Validate P2 buffer with Trib param
                                        .SBTTL  VALIDATE_P2_UCB,  Validate P2 buffer with UCB

                                ;++
                                ; VALIDATE_P2 - Validate P2 buffer parameters
                                ; VALIDATE_P2_TRIB - Validate P2 buffer with trib parameters
                                ; VALIDATE_P2_UCB - Validate P2 buffer with UCB
                                ;
                                ; This routine is called to validate the P2 buffer parameters. The parameters
                                ; are checked against a parameter table which verifies that the minimum value
                                ; and maximum value is not violated, and that invalid status flags are not set.
                                ;
                                ; Inputs:
                                ;
                                ;       R1 = Address of parameter verification table (VALIDATE_P2 entry only)
                                ;       R2 = Status word from UCB
                                ;       R3 = IRP address
                                ;       R4 = UCB or trib parameter block address for value
                                ;            checking (VALIDATE_P2 entry only)
                                ;       R5 = UCB address
                                ;
                                ;       IPL = FIPL
                                ;
                                ; Outputs:
                                ;
                                ;       R0 = status return of parameters
                                ;
                                ; If no error:
                                ;       R1 = Address of parameter verification table
                                ; If error:
                                ;       R1 = Bad parameter value
                                ;
                                ;       All other registers are preserved.
                                ;
                                ;--
                                        .ENABL  LSB
                                VALIDATE_P2_TRIB:                               ; Validate P2 buffer with TFB
                        54      DD      PUSHL   R4                              ; Save R4
                51      DD31    CF      MOVAB   TRIB_PARAM,R1                   ; Get address of verification table
                54      0134    C5      MOVAB   UCB$Z_XG_DDCMP(R5),R4           ; Get address of block to set
                                                                                ; DDCMP parameters
                        0A      11      BRB     10$                             ;
                                        .DSABL  LSB
                                VALIDATE_P2:                                    ; Validate P2 buffer parameters
                    03EE        8F      BB      PUSHR   #^M<R1,R2,R3,R5,R6,R7,R8,R9>    ; Save registers
                                                                                ; NB:R1 must be on top of stack
                  56    2C      A3      D0      MOVL    IRP$L_SVAPTE(R3),R6     ; Get system P2 buffer address
                                03      12      BNEQ    10$                     ; Br if a system buffer
```

```
                     0085   31  23E9  6182          BRW     150$                            ; Else, leave
                58     32 A3  3C  23EC  6183 10$:    MOVZWL  IRP$W_BCNT(R3),R8               ; Get size of P2 buffer
                   58    06  C6  23F0  6184          DIVL    #6,R8                           ; Calculate number of parameters
                   56    66  D0  23F3  6185          MOVL    P2B_L_POINTER(R6),R6            ; Point to start of P2 data
                               23F6  6186 ;
                               23F6  6187 ; Loop to check next parameter in P2 buffer
                               23F6  6188 ;
                   50    86  3C  23F6  6189 30$:    MOVZWL  (R6)+,R0                        ; Get parameter type from P2
                   55    86  D0  23F9  6190          MOVL    (R6)+,R5                        ; Get parameter value from P2
                   57    6E  D0  23FC  6191          MOVL    (SP),R7                         ; Get parameter table address
                               23FF  6192 ;
                               23FF  6193 ; Loop to check P2 buffer parameter to Line parameter table
                               23FF  6194 ;
                   59    87  B0  23FF  6195 40$:    MOVW    (R7)+,R9                        ; Get parameter + flags
                         72  13  2402  6196          BEQL    170$                            ; Br if end of verify table
         50        59    0C  00  ED  2404  6197          CMPZV   #PRM_V_TYPE,#PRM_S_TYPE,R9,R0   ; Parameters match?
                         16  13  2409  6198          BEQL    50$                             ; Br if yes
                         87  B5  240B  6199          TSTW    (R7)+                           ; Skip offset word
                             240D  6200          SKIP    PRM_V_MIN,R9,R7                 ; Skip minimum value
                             2413  6201          SKIP    PRM_V_MAX,R9,R7                 ; Skip maximum value
                             2419  6202          SKIP    PRM_V_INVALID,R9,R7             ; Skip invalid flags
                         DE  11  241F  6203          BRB     40$                             ; Try next parameter
                             2421  6204 ;
                             2421  6205 ; Match found - nullify if same value & check min,max,valid,invalid
                             2421  6206 ;
                   51    87  B0  2421  6207 50$:    MOVW    (R7)+,R1                        ; Get offset + width
                         54  D5  2424  6208          TSTL    R4                              ; Is data structure present?
                         2B  13  2426  6209          BEQL    100$                            ; Br if not - check values
         53        51    02  0E  EF  2428  6210          EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R1,R3 ; Get width only
         51        51    0E  00  EF  242D  6211          EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R1,R1 ; Get offset only
                   51    54  C0  2432  6212          ADDL    R4,R1                           ; Calculate address of datum
                             2435  6213          CASE    R3,TYPE=B,LIMIT=#1,<-           ; Br to handler
                             2435  6214                  60$,-                           ;   Byte value
                             2435  6215                  70$,-                           ;   Word value
                             2435  6216                  80$>                            ;   Longword value
                             243F  6217 ;
                             243F  6218 ; Byte value in structure
                             243F  6219 ;
               61    55    91  243F  6220 60$:    CMPB    R5,(R1)                         ; Is this the same?
                         08  11  2442  6221          BRB     90$                             ; Check result
                             2444  6222 ;
                             2444  6223 ; Word value
                             2444  6224 ;
               61    55    B1  2444  6225 70$:    CMPW    R5,(R1)                         ; Is this the same?
                         03  11  2447  6226          BRB     90$                             ; Check result
                             2449  6227
               61    55    D1  2449  6228 80$:    CMPL    R5,(R1)                         ; Is this the same?
                         05  12  244C  6229 90$:    BNEQ    100$                            ; Br if no - continue checks
                  FA     A6  B4  244E  6230          CLRW    -6(R6)                          ; Nullify the parameter code
                         1B  11  2451  6231          BRB     140$                            ; Try next parameter - skip checks
                             2453  6232
         05    59    0C  E1  2453  6233 100$:   BBC     #PRM_V_MIN,R9,110$              ; Br if no minimum value
                   87    55  B1  2457  6234          CMPW    R5,(R7)+                        ; Is the value too small?
                         1A  1F  245A  6235          BLSSU   170$                            ; Br if yes - error
         05    59    0D  E1  245C  6236 110$:   BBC     #PRM_V_MAX,R9,130$              ; Br if no maximum value
                   87    55  B1  2460  6237          CMPW    R5,(R7)+                        ; Is the value too big?
                         11  1A  2463  6238          BGTRU   170$                            ; Br if yes - error
```

```
                05 59    0E    E1  2465  6239 130$:   BBC     #PRM_V_INVALID,R9,140$          ; Br if no invalid flags
                   52    87    B3  2469  6240         BITW    (R7)+,R2                        ; Check invalid bits
                         08    12  246C  6241         BNEQ    170$                            ; Br if on - error
                   85 58    F5  246E  6242 140$:      SOBGTR  R8,30$                          ; Loop if more parameters
                                2471  6243
                   50    01    3C  2471  6244 150$:      MOVZWL  S^#SS$_NORMAL,R0                ; Set success return
                         06    11  2474  6245         BRB     180$                            ; And return
                                2476  6246
                   6E    50    3C  2476  6247 170$:      MOVZWL  R0,(SP)                         ; Return bad parameter code
                                2479  6248                                                    ; * R1 Must be on top of stack
                   50    14    3C  2479  6249         MOVZWL  #SS$_BADPARAM,R0                ; Set error return
                   03EE  8F    BA  247C  6250 180$:      POPR    #^M<R1,R2,R3,R5,R6,R7,R8,R9>    ; Restore registers
                               05  2480  6251         RSB                                     ; Return to caller
```

```
                        2481  6253          .SBTTL  UPDATE_P2,  Update UCB/TRIB based on P2 buffer parameters
                        2481  6254
                        2481  6255 ;++
                        2481  6256 ; UPDATE_P2 - Update UCB/TRIB with P2 buffer parameters
                        2481  6257 ;
                        2481  6258 ; This routine is called to update the UCB/TRIB with the P2 buffer parameters.
                        2481  6259 ; The parameters are stored in the appropriate cells of the UCB/TRIB.
                        2481  6260 ;
                        2481  6261 ; Inputs:
                        2481  6262 ;
                        2481  6263 ;       R1 = Address of parameter verification table
                        2481  6264 ;       R3 = IRP address
                        2481  6265 ;       R4 = UCB or ddcmp parameter block  address for storing
                        2481  6266 ;       R5 = UCB address
                        2481  6267 ;
                        2481  6268 ;       IPL = FIPL
                        2481  6269 ;
                        2481  6270 ; Outputs:
                        2481  6271 ;
                        2481  6272 ;       R0 = destroyed.
                        2481  6273 ;       All other registers are preserved.
                        2481  6274 ;
                        2481  6275 ;--
                        2481  6276
                        2481  6277 UPDATE_P2:                                          ; Update the UCB/DDCMP parametr
    03E6 8F     BB      2481  6278          PUSHR   #^M<R1,R2,R5,R6,R7,R8,R9>       ; Save registers
                        2485  6279                                                  ; NB:R1 must be on top of stack
56    2C A3     D0      2485  6280          MOVL    IRP$L_SVAPTE(R3),R6             ; Get system P2 buffer address
        62     13      2489  6281          BEQL    80$                             ; Br if no system buffer
58    32 A3     3C      248B  6282          MOVZWL  IRP$W_BCNT(R3),R8               ; Get size of P2 buffer
   58    06     C6      248F  6283          DIVL    #6,R8                           ; Calculate number of parameters
   56    66     D0      2492  6284          MOVL    P2B_L_POINTER(R6),R6            ; Point to start of data
                        2495  6285 ;
                        2495  6286 ; Loop to get next parameter from P2 buffer
                        2495  6287 ;
              50    86  3C  2495  6288 10$:     MOVZWL  (R6)+,R0                        ; Get parameter type from P2
              55    86  D0  2498  6289          MOVL    (R6)+,R5                        ; Get parameter value from P2
              57    6E  D0  249B  6290          MOVL    (SP),R7                         ; Get parameter table address
                        249E  6291 ;
                        249E  6292 ; Loop to store buffer parameter in UCB/DDCMP parameter block
                        249E  6293 ;
              59    87  B0  249E  6294 20$:     MOVW    (R7)+,R9                        ; Get parameter + flags
                    47  13  24A1  6295          BEQL    70$                             ; Br if end of verify table
51    59    0C    00  EF  24A3  6296          EXTZV   #PRM_V_TYPE,#PRM_S_TYPE,R9,R1   ; Get type field
              51    50  B1  24A8  6297          CMPW    R0,R1                           ; Parameters match?
                    16  13  24AB  6298          BEQL    30$                             ; Br if yes
                    87  B5  24AD  6299          TSTW    (R7)+                           ; Skip offset word
                        24AF  6300          SKIP    PRM_V_MIN,R9,R7                 ; Skip minimum value
                        24B5  6301          SKIP    PRM_V_MAX,R9,R7                 ; Skip maximum value
                        24BB  6302          SKIP    PRM_V_INVALID,R9,R7             ; Skip invalid flags
                    DB  11  24C1  6303          BRB     20$                             ; Try next parameter
                        24C3  6304 ;
                        24C3  6305 ; Match found - nullify if same value & check min,max,valid,invalid
                        24C3  6306 ;
              51    87  B0  24C3  6307 30$:     MOVW    (R7)+,R1                        ; Get offset + width
52    51    02    0E  EF  24C6  6308          EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R1,R2 ; Get width only
51    51    0E    00  EF  24CB  6309          EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R1,R1 ; Get offset only
```

```
                51    54    C0  24D0  6310          ADDL    R4,R1                           ; Calculate address of datum
                                24D3  6311          CASE    R2,TYPE=B,LIMIT=#1,<-           ; Br to handler
                                24D3  6312                  40$,-                           ;   Byte value
                                24D3  6313                  50$,-                           ;   Word value
                                24D3  6314                  60$>                            ;   Longword value
                                24DD  6315 ;
                                24DD  6316 ; Byte, word, longword value in structure
                                24DD  6317 ;
                61    55    90  24DD  6318 40$:     MOVB    R5,(R1)                         ; Store byte value
                      08    11  24E0  6319          BRB     70$                             ;
                61    55    B0  24E2  6320 50$:     MOVW    R5,(R1)                         ; Store word value
                      03    11  24E5  6321          BRB     70$                             ;
                61    55    D0  24E7  6322 60$:     MOVL    R5,(R1)                         ; Store longword value
                  A8  58    F5  24EA  6323 70$:     SOBGTR  R8,10$                          ; Br if more parameters
                                24ED  6324
                03E6  8F    BA  24ED  6325 80$:     POPR    #^M<R1,R2,R5,R6,R7,R8,R9>       ; Restore registers
                                05  24F1  6326          RSB                             ; Return to caller
```

```
                        24F2  6328          .SBTTL  RETURN_P2,  Return UCB/DDCMP buffer parameters
                        24F2  6329
                        24F2  6330 ;++
                        24F2  6331 ; RETURN_P2 - Return P2 buffer parameters
                        24F2  6332 ;
                        24F2  6333 ; This routine is called to return the UCB/DDCMP buffer parameters.
                        24F2  6334 ;
                        24F2  6335 ; Inputs:
                        24F2  6336 ;
                        24F2  6337 ;       R1 = Address of return table (same format as verification table)
                        24F2  6338 ;       R2 = Address of user buffer in which to return the parameters
                        24F2  6339 ;       R3 = IRP address
                        24F2  6340 ;       R4 = UCB or DDCMP paramter block address for storing
                        24F2  6341 ;       R5 = UCB address
                        24F2  6342 ;       IRP$W_BCNT(R3) = Size of transfer
                        24F2  6343 ;
                        24F2  6344 ; Outputs:
                        24F2  6345 ;
                        24F2  6346 ;       R0 = destroyed.
                        24F2  6347 ;       All other registers are preserved.
                        24F2  6348 ;
                        24F2  6349 ;--
                        24F2  6350
                        24F2  6351 RETURN_P2:                                   ; Return P2 buffer parameters
    01E6 8F     BB      24F2  6352          PUSHR   #^M<R1,R2,R5,R6,R7,R8>      ; Save registers
    56   52     D0      24F6  6353          MOVL    R2,R6                       ; Get user buffer address
         50     13      24F9  6354          BEQL    60$                         ; Br if no system buffer
58     32 A3     3C     24FB  6355          MOVZWL  IRP$W_BCNT(R3),R8           ; Get size of buffer
    58   06     C6      24FF  6356          DIVL    #6,R8                       ; Calculate the number of param
                        2502  6357 ;
                        2502  6358 ; Loop to return next parameter
                        2502  6359 ;
         55     81  B0  2502  6360 10$:     MOVW    (R1)+,R5                    ; Get parameter + flags
                44  13  2505  6361          BEQL    60$                         ; Br if end of verify table
57   55   0C     00  EF  2507  6362          EXTZV   #PRM_V_TYPE,#PRM_S_TYPE,R5,R7 ; Get type field
             86 57  B0  250C  6363          MOVW    R7,(R6)+                    ; Return parameter
             57 81  B0  250F  6364          MOVW    (R1)+,R7                    ; Get offset + width
52   57      02  0E  EF  2512  6365          EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R7,R2 ; Get width only
57   57      0E  00  EF  2517  6366          EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R7,R7 ; Get offset only
             57 54  C0  251C  6367          ADDL    R4,R7                       ; Calculate address of datum
                        251F  6368          CASE    R2,TYPE=B,LIMIT=#1,<-       ; Br to handler
                        251F  6369                  20$,-                       ;   Byte value
                        251F  6370                  30$,-                       ;   Word value
                        251F  6371                  40$>                        ;   Longword value
                        2529  6372 ;
                        2529  6373 ; Byte, word, longword value in structure
                        2529  6374 ;
             86 67  9A  2529  6375 20$       MOVZBL  (R7),(R6)+                  ; Store byte value
                08  11  252C  6376          BRB     50$                         ;
             86 67  3C  252E  6377 30$       MOVZWL  (R7),(R6)+                  ; Store word value
                03  11  2531  6378          BRB     50$                         ;
             86 67  D0  2533  6379 40$       MOVL    (R7),(R6)+                  ; Store longword value
                        2536  6380 50$       SKIP    PRM_V_MIN,R5,R1             ; Skip minimum value
                        253C  6381           SKIP    PRM_V_MAX,R5,R1             ; Skip maximum value
                        2542  6382           SKIP    PRM_V_INVALID,R5,R1         ; Skip invalid flags
          B7 58  F5     2548  6383          SOBGTR  R8,10$                      ; Try for more parameters
                        254B  6384
```

```
01E6 8F   BA  254B  6385 60$:    POPR    #^M<R1,R2 R5,R6,R7,R8>          ; Restore registers
          05  254F  6386          RSB                                     ; Return to caller
```

```
                                2550  6388          .SBTTL  UNPACK_P2_BUF,  Unpack a P2 parameter from P2 buffer
                                2550  6389
                                2550  6390 ;++
                                2550  6391 ; UNPACK_P2_BUF - Unpack a P2 parameter from P2 buffer
                                2550  6392 ;
                                2550  6393 ; Functional description:
                                2550  6394 ;
                                2550  6395 ; This routine is called to get a P2 parameter from the P2 buffer.
                                2550  6396 ;
                                2550  6397 ; Inputs:
                                2550  6398 ;
                                2550  6399 ;       R1 = Parameter type code
                                2550  6400 ;       R3 = IRP address
                                2550  6401 ;       R5 = UCB address
                                2550  6402 ;
                                2550  6403 ; Outputs:
                                2550  6404 ;
                                2550  6405 ;       R0 = Low bit set if specified Parameter type code is found in P2
                                2550  6406 ;       R2 = Parameter value if success else destroyed
                                2550  6407 ;
                                2550  6408 ;       All other registers are preserved.
                                2550  6409 ;
                                2550  6410 ;--
                                2550  6411
                                2550  6412 UNPACK_P2_BU .                                  ; Unpack P2 buffer
    00E0 8F     BB              2550  6413          PUSHR   #^M<R5,R6,R7>                  ; Save registers
56    2C A3     D0              2554  6414          MOVL    IRP$L_SVAPTE(R3),R6            ; Get system P2 buffer address
          1C    13              2558  6415          BEQL    20$                            ; Br if none
57    32 A3     3C              255A  6416          MOVZWL  IRP$W_BCNT(R3),R7              ; Get size of P2 buffer
   57     06     C6              255E  6417          DIVL    #6,R7                          ; Caculate number of parameters
56    0C A6     9E              2561  6418          MOVAB   P2B_I_DATA(R6),R6              ; Point to start of data
   50     01     3C              2565  6419          MOVZWL  S^#SS$_NORMAL,R0               ; Assume success
                                2568  6420 ;
                                2568  6421 ; Loop to check next parameter in P2 buffer
                                2568  6422 ;
55    86     3C                 2568  6423 10$:     MOVZWL  (R6)+,R5                       ; Get parameter type from P2
52    86     D0                 256B  6424          MOVL    (R6)+,R2                       ; Get parameter value from P2
55    51     B1                 256E  6425          CMPW    R1,R5                          ; Parameters match?
          05     13              2571  6426          BEQL    30$                            ; Br if yes
   F2 57     F5                 2573  6427          SOBGTR  R7,10$                         ; Br if more parameters
                                2576  6428
      50     D4                 2576  6429 20$:     CLRL    R0                             ; Return error
00E0  8F     BA                 2578  6430 30$:     POPR    #^M<R5,R6,R7>                  ; Restore registers
          05     257C  6431          RSB                                    ; Return to caller
                                257D  6432
                                257D  6433 XG_END:
                                257D  6434          .END
```

```
$$$                             = 00000020 R      02
$$$OFF                          = 00000009
$$$TYP                          = 00005479
$$$WID                          = 00004000
$$OP                            = 00000002
ABORTIO                           0000051A R      03
ACB$L_KAST                      = 00000018
ACCESS                            00000B07 R      03
ADDFREELIST                       00001627 R      03
ALLOC_P2BUF                       00000B20 R      03
ALLOC_PROT_BUFFER                 000007A6 R      03
ALT_CLEANFDT                      00000403 R      03
ALT_ENTRY                         0000035E R      03
ALT_ENTRY_BISYNC                  000003F3 R      03
ALT_ENTRY_LAPB                    000003F3 R      03
ALT_RCVFDT                        000003D9 R      03
ATS_UBA                           ******** R    X 02
AWAIT_CONT                        000011E2 R      03
AWAIT_UNIT                        000011CB R      03
BCHAR_NPOS                        00001EF4 R      03
BCHAR_NPOS_CMP                    00001F0C R      03
BCHAR_PPOS                        00001EFF R      03
BCHAR_PPOS_CMP                    00001F16 R      03
BISYNC$A_RCV_BUFFER               0000001C
BISYNC$B_XQCNT                    00000010
BISYNC$C_HDRLEN                   0000000C
BISYNC$C_LENGTH                   00000074
BISYNC$K_HDRLEN                   0000000C
BISYNC$K_LENGTH                   00000074
BISYNC$L_HDR                      00000000
BISYNC$L_TQE                      00000038
BISYNC$Q_BLANK                    00000020
BISYNC$Q_CLEANQ                   00000028
BISYNC$Q_INIT_STATE_INFO          00000014
BISYNC$Q_XMTQ                     00000030
BISYNC$V_RCV_COMPLETE           = 00000000
BISYNC$V_TIMER_RUNNING          = 00000001
BISYNC$W_DROP_RCV                 00000011
BISYNC$W_RCV_INDEX                0000000C
BISYNC$W_STATUS                   0000000E
BISYNC_TIMER                      00001E18 R      03
BUG$_NOBUFPCKT                    ******** R    X 03
CANCEL                            000020D0 R      03
CHECK_BUFS                        00000AE1 R      03
CHECK_NEXT                        00001ECD R      03
CHECK_P1                          00000B0D R      03
CHECK_P2                          00000AE3 R      03
CHECK_RCV                         00001E2D R      03
CHECK_XMT                         00001FA7 R      03
CHG_TRIB                          00000C48 R      03
CHG_UCB                           00000B78 R      03
CLEAN                             00000F54 R      03
CLEANFDT                          00000412 R      03
CLEAN_BISYNC                      00001021 R      03
CLEAN_FORK_ENTRY                  00000F9A R      03
COM$DELATTNAST                    ******** R    X 03
COM$DRVDEALMEM                    ******** R    X 03
COM$FLUSHATTNS                    ******** R    X 03
COM$POST                          ******** R    X 03
COM$SETATTNAST                    ******** R    X 03
COM_RCVFDT                        00000522 R      03
COM_XMITBISYNC                    00000432 R      03
COM_XMITFDT                       000002ED R      03
COM_XMITLAPB                      00000432 R      03
CONTROL_INIT                      0000014E R      03
COP_BUFF                          00000342 R      03
CRB$L_INTD                      = 00000024
CRB$L_INTD2                     = 00000048
CTSRET                            00000D63 R      03
CTS_TIMER                         00001D30 R      03
CXB$C_TRAILER                   = 00000004
CXB$K_HEADER                    = 00000048
DCS_SCOM                          ******** R    X 02
DDB$L_DDT                       = 0000000C
DDCMP                             ******** R    X 03
DDCMP$C_HEADER                  = 00000006
DDCMP$C_SIZEOFQ                 = 0000007F
DEF_BISYNC_PARAM                  00000146 RG     03
DEF_BISYNC_PARAMSZ              = 00000008
DEF_LAPB_PARAM                    0000013E RG     03
DEF_LAPB_PARAMSZ                = 00000008
DEF_LINE_PARAM                    00000122 RG     03
DEF_LINE_PARAMSZ                = 00000014
DEF_SYNC_PARAM                    00000136 RG     03
DEF_SYNC_PARAMSZ                = 00000008
DEF_TRIB_PARAM                    00000118 RG     03
DEF_TRIB_PARAMSZ                = 0000000A
DEV$M_AVC                       = 00040000
DEV$M_IDV                       = 04000000
DEV$M_NET                       = 00002000
DEV$M_ODV                       = 08000000
DEVTIMER                          00001D77 R      03
DEVTIMER_ALT                      00001DF5 R      03
DLA$A_POSTQ                       00000004
DLA$A_XMT_INPR                    00000000
DLA$C_ADDR_LENGTH                 00000008
DLA$K_ADDR_LENGTH                 00000008
DLK$B_MAINT                       00000008
DLK$B_MAXREP                      00000002
DLK$B_MAXSEL                      00000003
DLK$B_MRB                         00000009
DLK$B_MSGCNT                      00000001
DLK$B_TRIB                        00000000
DLK$C_ACTNOTCOM                 = 00000009
DLK$C_CHAR                      = 00000005
DLK$C_RCVMSG                    = 00000001
DLK$C_REQEBA                    = 00000003
DLK$C_START_TIMER               = 00000006
DLK$C_STOP_TIMER                = 00000007
DLK$C_USRINT                    = 00000004
DLK$C_XMTMSG                    = 00000002
DLK$M_CLEAR                     = 00000001
DLK$M_DATACRC                   = 00000004
DLK$M_GLOB                      = 00000004
```

```
DLK$M_HDRCRC            = 00000002
DLK$M_HDRERR            = 00000008
DLK$M_MAINT             = 00000004
DLK$M_MNTRCV            = 00000080
DLK$M_MSGCNT            = 00000010
DLK$M_MSGSENT           = 00000001
DLK$M_PRSTERR           = 00000002
DLK$M_QEMPTY            = 00000002
DLK$M_QFULERR           = 00000020
DLK$M_RCVOVR            = 00000020
DLK$M_REPTIM            = 00000080
DLK$M_REPWAIT           = 00000100
DLK$M_SELTIM            = 00000020
DLK$M_SELWAIT           = 00000040
DLK$M_SETDEF            = 00000800
DLK$M_START             = 00000001
DLK$M_STOP              = 00000002
DLK$M_STRTRCV           = 00000040
DLK$M_TMREXPD           = 00008000
DLK$M_TRIB              = 00000002
DLK$M_XMTACK            = 00000200
DLK$V_DUPLEX            = 00000002
DLK$V_MNTRCV            = 00000007
DLK$V_MSGSENT           = 00000000
DLK$V_PRSTERR           = 00000001
DLK$V_RCVACK            = 00000008
DLK$V_SETDEF            = 0000000B
DLK$V_STATYP            = 00000000
DLK$V_TMREXPD           = 0000000F
DLK$V_TRNLK             = 0000000E
DLK$V_XMTCMP            = 0000000A
DLK$V_XMTERR            = 00000004
DLK$W_REPWAIT             00000004
DLK$W_SELWAIT             00000006
DMF_CSR                 = FFFFFFFC
DPT$B_FLAGS             = 0000000D
DPT$C_LENGTH            = 00000038
DPT$C_VERSION           = 00000004
DPT$INITAB                00000038 R      02
DPT$M_NOUNLOAD          = 00000004
DPT$REINITAB              00000054 R      02
DPT$TAB                   00000000 R      02
DRIVER_END                ******** X    02
DRPCTSRET                 00001900 R      03
DSC$A_POINTER           = 00000004
DT$_DMF32                 ******** X    02
DYN$C_BUFIO             = 00000013
DYN$C_CRB               = 00000005
DYN$C_DDB               = 00000006
DYN$C_DPT               = 0000001E
DYN$C_NET               = 00000017
DYN$C_UCB               = 00000010
EXE$ABORTIO               ******** X    03
EXE$ALLOCBUF              ******** X    03
EXE$ALONONPAGED           ******** X    03
EXE$BUFFRQUOTA            ******** X    03
EXE$BUFQUOPRC             ******** X    03
EXE$FINISHIO              ******** X    03
EXE$FORK                  ******** X    03
EXE$GL_ABSTIM             ******** X    03
EXE$GL_TENUSEC            ******** X    03
EXE$GL_UBDELAY            ******** X    03
EXE$GQ_SYSTIME            ******** X    03
EXE$INSTIMQ               ******** X    03
EXE$QIORETURN             ******** X    03
EXE$READCHK               ******** X    03
EXE$RMVTIMQ               ******** X    03
EXE$WRITECHK              ******** X    03
EXE$WRITECHKR             ******** X    03
FILLFREELIST              0000161F R      03
FINISH_RCV_IO             00001C54 R      03
FINISH_RCV_IO_BISYNC      00001C5C R      03
FINISH_RCV_IO_LAPB        00001C5C R      03
FINISH_XMT_IO             00001CF5 R      03
FORK_DONE                 00001A20 R      03
FORK_DONE_BISYNC          00001B39 R      03
FORK_DONE_LAPB            00001AA5 R      03
FUNCTABLE                 00000038 R      03
FUNCTAB_LEN             = 0000004C
GET_CHAR_BUFS             00000A7C R      03
GF$B_BABTIM               00000016
GF$B_CURSEL               00000008
GF$B_DIPL                 0000000A
GF$B_DRVCHR               00000002
GF$B_FIPL                 00000009
GF$B_GH_CRC               00000029
GF$B_LSE                  00000023
GF$B_MAXSEL               00000007
GF$B_MDF_CRC              0000002A
GF$B_RSE                  00000028
GF$B_STATE                00000000
GF$B_STRTIM               00000015
GF$B_STRTSEL              0000002B
GF$B_TIMER_STATE          00000001
GF$C_ERREND               00000029
GF$C_ERRSRT               0000001F
GF$C_LENGTH               0000002C
GF$K_ERREND               00000029
GF$K_ERRSRT               0000001F
GF$K_LENGTH               0000002C
GF$L_BABTMR               0000001B
GF$L_SELEND               0000000D
GF$L_STRTMR               00000017
GF$L_TQE_STS              00000011
GF$W_GEB                  0000000B
GF$W_LSETYP               0000001F
GF$W_LSE_BCTRS            00000021
GF$W_NEXTCNT              00000005
GF$W_RSETYP               00000024
GF$W_RSE_BCTRS            00000026
GF$W_SELQAI               00000003
IDB$B_COMBO_CSR_OFFSET  = 0000000F
IDB$B_COMBO_VECTOR_OFFSET = 00000010
IDB$B_VECTOR            = 0000000B
```

```
IDB$L_UCBLST                    = 00000018
IGN_CHAR                          00001EFA R      03
IGN_CHAR_CMP                      00001F14 R      03
ILLEGAL                           00001EF0 R      03
INIT_PARAM                        000001E9 R      03
INTERR                            00001A05 R      03
INTEXIT                           000019FB R      03
INT_VEC                         = 00000004
IO$M_CLR_COUNT                  = 00000400
IO$M_RD_COUNT                   = 00000100
IO$V_ATTNAST                    = 00000008
IO$V_CLR_COUNT                  = 0000000A
IO$V_CTRL                       = 00000009
IO$V_NOW                        = 00000006
IO$V_RD_COUNT                   = 00000008
IO$V_RD_MODEM                   = 00000007
IO$V_SHUTDOWN                   = 00000007
IO$V_STARTUP                    = 00000006
IO$_CLEAN                       = 0000001E
IO$_READLBLK                    = 00000021
IO$_READPBLK                    = 0000000C
IO$_READVBLK                    = 00000031
IO$_SENSEMODE                   = 00000027
IO$_SETCHAR                     = 0000001A
IO$_SETMODE                     = 00000023
IO$_VIRTUAL                     = 0000003F
IO$_WRITELBLK                   = 00000020
IO$_WRITEPBLK                   = 0000000B
IO$_WRITEVBLK                   = 00000030
IOC$ALOUBAMAP                     ********    X   03
IOC$INITIATE                      ********    X   03
IOC$LOADUBAMAPA                   ********    X   03
IOC$MNTVER                        ********    X   03
IOC$RELMAPREG                     ********    X   03
IOC$REQCOM                        ********    X   03
IOC$RETURN                        ********    X   03
IOC$WFIKPCH                       ********    X   03
IO_DONE                           00001CAB R      03
IPL$_POWER                      = 0000001F
IPL$_QUEUEAST                   = 00000006
IPL$_SYNCH                      = 00000008
IPL$_TIMER                      = 00000008
IRP$B_TYPE                      = 0000000A
IRP$B_XGFUNC                      00000021
IRP$L_ABCNT                     = 00000040
IRP$L_DIAGBUF                   = 0000004C
IRP$L_MEDIA                     = 00000038
IRP$L_PID                       = 0000000C
IRP$L_SVAPTE                    = 0000002C
IRP$Q_STATION                   = 00000040
IRP$V_DIAGBUF                   = 00000007
IRP$V_FUNC                      = 00000001
IRP$W_BCNT                      = 00000032
IRP$W_BOFF                      = 00000030
IRP$W_CHAN                      = 00000028
IRP$W_FUNC                      = 00000020
IRP$W_QUOTA                       00000040
IRP$W_STS                       = 0000002A
JIB$L_BYTCNT                    = 00000020
LAPB$B_DEI                        00000016
LAPB$B_XQCNT                      00000010
LAPB$C_ERREND                     00000017
LAPB$C_ERRSTRT                    00000012
LAPB$C_HDRLEN                     0000000C
LAPB$C_LENGTH                     00000038
LAPB$K_ERREND                     00000017
LAPB$K_ERRSTRT                    00000012
LAPB$K_HDRLEN                     0000000C
LAPB$K_LENGTH                     00000038
LAPB$L_HDR                        00000000
LAPB$Q_BLANK                      00000030
LAPB$Q_CLEANQ                     00000028
LAPB$Q_XMTQ                       00000020
LAPB$W_DEIBC                      00000014
LAPB$W_DEITYP                     00000012
LINE_PARAM                        00000084 R      03
LINE_PRM_BUFSIZ                 = 0000005A
LOAD_RCV_MPR                      00001781 R      03
LOAD_XMT_MPR                      00000D38 R      03
MASKH                           = 00000000
MASKL                           = 40000000
MAX_RCVS                        = 00000004
MFD$B_ADDR                        00000005
MFD$B_MSGID                       00000000
MFD$B_NUMB                        00000004
MFD$B_RESP                        00000003
MFD$C_LENGTH                      00000006
MFD$K_LENGTH                      00000006
MFD$W_CNTFLG                      00000001
MFD$W_TYPFLG                      00000001
MMG$GL_SPTBASE                    ********    X   03
NMA$C_CTCIR_DEI                 = 000003FC
NMA$C_DPX_FUL                   = 00000000
NMA$C_DPX_HAL                   = 00000001
NMA$C_LINCN_LOO                 = 00000001
NMA$C_LINCN_NOR                 = 00000000
NMA$C_LINPR_BSY                 = 00000009
NMA$C_LINPR_CON                 = 00000001
NMA$C_LINPR_DMC                 = 00000004
NMA$C_LINPR_LAPB                = 00000005
NMA$C_LINPR_POI                 = 00000000
NMA$C_LINPR_TRI                 = 00000002
NMA$C_PCCI_ARB                  = 00000479
NMA$C_PCCI_MST                  = 00000AFA
NMA$C_PCCI_MTR                  = 0000047A
NMA$C_PCCI_TRI                  = 00000474
NMA$C_PCLI_BFN                  = 00000451
NMA$C_PCLI_BPC                  = 00000B37
NMA$C_PCLI_BUS                  = 00000AF1
NMA$C_PCLI_CON                  = 00000456
NMA$C_PCLI_DUP                  = 00000457
NMA$C_PCLI_FRA                  = 00000B31
NMA$C_PCLI_INTL3                = 00000B30
NMA$C_PCLI_MCL                  = 00000B35
```

NMA$C_PCLI_MNTL	=	00000B2C			RCV_Z_HEADER		00000042			
NMA$C_PCLI_NMS	=	00000AFA			READ_MODEM		00000EEC	R		03
NMA$C_PCLI_PRO	=	00000458			RECEIVE_DONE		00001B84	R		03
NMA$C_PCLI_RTT	=	00000461			RECEIVE_INTR		00001914	R		03
NMA$C_PCLI_STI1	=	00000B32			REGDUMP		00002042	R		03
NMA$C_PCLI_STI2	=	00000B33			RETURN_P2		000024F2	R		03
NMA$C_PCLI_SYC	=	00000B36			SCH$GL_PCBVEC		********		X	03
NMA$C_PCLI_TMO	=	00000B34			SCHED_FORK		00001A0A	R		03
NMA$C_STATE_OFF	=	00000001			SENSEMODEFDT		00000828	R		03
NMA$C_STATE_ON	=	00000000			SENSEMODE_CTRL		0000090E	R		03
NUM_MAP_REG	=	00000004			SENSE_MODEM		00000A4D	R		03
OFF_M_VALUE	=	00003FFF			SETMODEFDT		0000056A	R		03
OFF_S_VALUE	=	0000000E			SETMODE_CTRL		0000061F	R		03
OFF_S_WIDTH	=	00000002			SET_CHAR		00001440	R		03
OFF_V_VALUE	=	00000000			SET_DEFAULT_CHAR		00000C03	R		03
OFF_V_WIDTH	=	0000000E			SET_UNIT_DDCMP		000013C4	R		03
P1	=	00000000			SET_UNIT_LAPB_COMP		000013B2	R		03
P2	=	00000004			SET_UNIT_LAPB_ERROR		000013AA	R		03
P2B_B_SPARE		0000000B			SHUTDOWN_CIRCUIT		000021D6	R		03
P2B_B_TYPE		0000000A			SHUTDOWN_LINE		000020FE	R		03
P2B_C_LENGTH		0000000C			SHUTDOWN_LINE_ALT		00002100	R		03
P2B_L_BUFFER		00000004			SIZ...	=	00000002			
P2B_L_POINTER		00000000			SS$_ABORT	=	0000002C			
P2B_T_DATA		0000000C			SS$_ACCVIO	=	0000000C			
P2B_W_SIZE		00000008			SS$_BADPARAM	=	00000014			
P3	=	00000008			SS$_BUFFEROVF	=	00000601			
P4	=	0000000C			SS$_CTRLERR	=	00000054			
P5	=	00000010			SS$_DEVACTIVE	=	000002C4			
PCB$L_JIB	=	00000080			SS$_DEVINACT	=	000020D4			
PCB$L_PID	=	00000060			SS$_ENDOFFILE	=	00000870			
PCB$Q_PRIV	=	00000084			SS$_NOPRIV	=	00000024			
POKE_USER		0000205B	R	03	SS$_NORMAL	=	00000001			
PR$_IPL	=	00000012			SS$_POWERFAIL	=	00000364			
PRM_M_INVALID	=	00004000			SS$_TIMEOUT	=	0000022C			
PRM_M_MAX	=	00002000			STARTIO		00000C7D	R		03
PRM_M_MIN	=	00001000			START_BISYNC_TIMER		000015C9	R		03
PRM_M_TYPE	=	00000FFF			START_CIRCUIT		00001106	R		03
PRM_S_TYPE	=	0000000C			START_CIRCUIT_COMP		000011C2	R		03
PRM_V_INVALID	=	0000000E			START_CIRCUIT_ERROR		000011BA	R		03
PRM_V_MAX	=	0000000D			START_CTRL_ERROR		000010F4	R		03
PRM_V_MIN	=	0000000C			START_DISP		00000C92	R		03
PRM_V_TYPE	=	00000000			START_LINE		00001055	R		03
PRV$V_CMKRNL	=	00000000			START_POWERFAIL		000011B5	R		03
QUEPKT		00000A6C	R	03	START_RECEIVE		00001702	R		03
RCVFDT		000004F2	R	03	START_RECEIVE_BISYNC		00001696	R		03
RCVLAPB		000003D9	R	03	START_TIMER		00000EC9	R		03
RCV_B_BLKTYPE		0000000A			START_TRANSMIT		00000CA9	R		03
RCV_B_FIPL		0000000B			START_UNIT_BISYNC		0000148E	R		03
RCV_B_SLT		00000014			START_UNIT_BISYNC_COMP		000015B7	R		03
RCV_CXB_SPARE		00000015			START_UNIT_BISYNC_ERROR		000015AF	R		03
RCV_ERROR		00001EB2	R	03	START_UNIT_LAPB		0000129E	R		03
RCV_L_BACC		00000010			STOP_BISYNC_TIMER		000020A5	R		03
RCV_L_LINK		00000000			STOP_RCV_BISYNC		0000200C	R		03
RCV_T_DATA		00000048			TF$A_BUFPTR		00000048			
RCV_W_BLKSIZE		00000008			TF$A_DEV_TIMER		00000050			
RCV_W_ERROR		0000000E			TF$A_GFB		0000004C			
RCV_W_MSGSIZ		0000000C			TF$A_POSTQ		00000058			

```
TF$A_XMT_INPR           00000054        TF$W_LBETYP                     000001B0
TF$B_A                  00000013        TF$W_LRTOTYP                    000001BF
TF$B_ADDR               0000000D        TF$W_RBEBC                      000001B7
TF$B_CHAR               0000000C        TF$W_RBETYP                     000001B5
TF$B_CURRTO             0000001B        TF$W_REPWAI                     00000018
TF$B_DEI                000001AF        TF$W_RRTOTYP                    000001C2
TF$B_DEO                000001AA        TF$W_SELSP                      000001A4
TF$B_LBE                000001B4        TF$W_SELTYP                     000001A2
TF$B_LRTO               000001C1        TF$W_STOBC                      000001BC
TF$B_MAXRTO             0000001A        TF$W_STOTYP                     000001BA
TF$B_MMCTR              0000001D        TF$W_TEB                        0000001E
TF$B_MSGCNT             0000001C        TF$W_THRES                      000001C5
TF$B_N                  00000012        TIMEOUT                         0000208D R      03
TF$B_NAKRSN             00000188        TIMER_END                       00001F70 R      03
TF$B_R                  00000011        TIMER_END_CHKCMP                00001F75 R      03
TF$B_RADDR              0000000F        TIMER_END_CHKCMP_ALT            00001F7A R      03
TF$B_RBE                000001B9        TIMER_END_COMP                  00001F20 R      03
TF$B_REPTIM             00000017        TIMER_END_UPD                   00001F62 R      03
TF$B_RRTO               000001C4        TQE$B_RMOD                    = 00000028
TF$B_SELTIM             00000016        TQE$B_RQTYPE                  = 0000000B
TF$B_SFLAGS             00000189        TQE$C_SSREPT                  = 00000005
TF$B_STO                000001BE        TQE$C_SSSNGL                  = 00000001
TF$B_T                  00000014        TQE$L_FPC                     = 0000000C
TF$B_X                  00000015        TQE$L_FR3                     = 00000010
TF$B_XADDR              0000000E        TQE$L_FR4                     = 00000014
TF$B_XQCNT              00000010        TQE$Q_DELTA                   = 00000020
TF$C_ERREND             000001C5        TRANSMIT_DONE                   00001CD4 R      03
TF$C_ERRSTRT            0000018A        TRANSMIT_INTR                   000017F4 R      03
TF$C_LENGTH             000001C8        TRANSMIT_IO_DONE                00001C99 R      03
TF$K_ERREND             000001C5        TRIB_PARAM                      000000F8 R      03
TF$K_ERRSTRT            0000018A        TRIB_PRM_BUFSIZ               = 00000018
TF$K_LENGTH             000001C8        UBA$C_MAP                     = 00000800
TF$L_DBYTR              0000018C        UBA$M_MAP_VALID               = 80000000
TF$L_DBYTX              00000192        UCB$A_XG_CLEAN                  00000168
TF$L_DMSGR              00000198        UCB$A_XG_FRAME_ADDR             00000158
TF$L_DMSGX              0000019E        UCB$A_XG_PRO_BUFFER             000000D0
TF$L_HDR                00000000        UCB$B_DEVCLASS                = 00000040
TF$L_QACK               0000005C        UCB$B_DEVTYPE                 = 00000041
TF$L_QNAK               0000008C        UCB$B_DIPL                    = 0000005E
TF$L_QREP               000000BC        UCB$B_FIPL                    = 0000000B
TF$L_QSTACK             0000011C        UCB$B_TYPE                    = 0000000A
TF$L_QSTRT              000000EC        UCB$B_XG_BFN                    00000153
TF$L_TQE                0000014C        UCB$B_XG_CON                    00000152
TF$Q_CMPQ               00000028        UCB$B_XG_COUNT                  0000010F
TF$Q_CTLQ               00000020        UCB$B_XG_DUP                    00000151
TF$Q_RTOQ               00000038        UCB$B_XG_INUS                   0000010E
TF$Q_XMTOVF             00000040        UCB$B_XG_MNT_LOOPB              00000157
TF$Q_XMTQ               00000030        UCB$B_XG_MODEM_CLR              00000156
TF$W_DBRTYP             0000018A        UCB$B_XG_PRO                    00000150
TF$W_DBXTYP             00000190        UCB$B_XG_PROTYPE                00000167
TF$W_DEIBC              000001AD        UCB$B_XG_SETPRM                 00000150
TF$W_DEITYP             000001AB        UCB$B_XG_SPD                    00000154
TF$W_DEOBC              000001A8        UCB$B_XG_TIMEOUT                00000155
TF$W_DEOTYP             000001A6        UCB$B_XG_WFCTS_SEC              00000121
TF$W_DMRTYP             00000196        UCB$B_XG_XMTCNT                 00000164
TF$W_DMXTYP             0000019C        UCB$B_XG_XSTATE                 00000120
TF$W_LBEBC              000001B2        UCB$C_LENGTH                  = 00000090
```

```
UCB$C_XG_LENGTH             =  00000174
UCB$L_CRB                   =  00000024
UCB$L_DEVCHAR               =  00000038
UCB$L_DEVDEPEND             =  00000044
UCB$L_FR4                   =  00000014
UCB$L_IRP                   =  00000058
UCB$L_SVAPTE                =  00000078
UCB$L_XG_AST                   00000090
UCB$L_XG_PID                   0000012C
UCB$L_XG_RCV_INPR           =  000000EC
UCB$L_XG_STATE_INFO1        =  0000015C
UCB$L_XG_STATE_INFO2        =  00000160
UCB$L_XG_TQE                   00000094
UCB$L_XG_XMTENQ                00000104
UCB$L_XG_XMT_TIMEOUT           00000108
UCB$M_INT                   =  00000002
UCB$M_ONLINE                =  00000010
UCB$M_POWER                 =  00000020
UCB$M_TIM                   =  00000001
UCB$Q_XG_ATTN                  000000D4
UCB$Q_XG_FREE                  000000E4
UCB$Q_XG_POST                  000000F4
UCB$Q_XG_RCVS                  000000DC
UCB$Q_XG_RCV_INPR              000000EC
UCB$Q_XG_STATE_INFO            0000015C
UCB$V_ONLINE                =  00000004
UCB$V_POWER                 =  00000005
UCB$V_TIMOUT                =  00000006
UCB$W_BCNT                  =  0000007E
UCB$W_BOFF                  =  0000007C
UCB$W_DEVBUFSIZ             =  00000042
UCB$W_DEVSTS                =  00000068
UCB$W_ERRCNT                =  00000082
UCB$W_REFC                  =  0000005C
UCB$W_SIZE                  =  00000008
UCB$W_STS                   =  00000064
UCB$W_XG_CHANC                 00000132
UCB$W_XG_CHANL                 00000130
UCB$W_XG_DSC                   0000012A
UCB$W_XG_MFDLEN                00000165
UCB$W_XG_QUOTA                 0000010C
UCB$W_XG_RCVCSR                00000122
UCB$W_XG_RCVERR                00000126
UCB$W_XG_RCV_INIT           =  000000F2
UCB$W_XG_RCV_INPR_INDX      =  000000F0
UCB$W_XG_XMTCSR                00000124
UCB$W_XG_XMTERR                00000128
UCB$Z_XG_DDCMP                 00000134
UCB$Z_XG_DLA_ADDR              0000016C
UCB$Z_XG_SYNC                  00000144
UCB$Z_XG_VECTOR                00000110
UCB$Z_XG_XMT_INPR              000000FC
UNIT_INIT                      0000015A  R      03
UNPACK_P2_BUF                  00002550  R      03
UPDATE_P2                      00002481  R      03
VA$M_BYTE                   =  000001FF
VA$S_VPN                    =  00000015
VA$V_VPN                    =  00000009
VALIDATE_P2                    000023DF  R      03
VALIDATE_P2_TRIB               000023C1  R      03
VALIDATE_P2_UCB                000023CF  R      03
VEC$B_DATAPATH              =  00000013
VEC$B_NUMREG                =  00000012
VEC$L_ADP                   =  00000014
VEC$L_IDB                   =  00000008
VEC$L_INITIAL               =  0000000C
VEC$L_UNITINIT              =  00000018
VEC$S_MAPREG                =  0000000F
VEC$V_MAPREG                =  00000000
VEC$W_MAPREG                =  00000010
XG$B_BAUD                   =  00000004
XG$B_BPC                    =  00000008
XG$B_ERR_CNTRL              =  00000000
XG$B_ICLK                   =  00000007
XG$B_MNTLOOP                =  00000009
XG$B_NUM_SYNC               =  00000005
XG$B_PROTOCOL               =  00000001
XG$B_RX_BPC                 =  00000003
XG$B_SYNC_REG               =  00000006
XG$B_TX_BPC                 =  00000002
XG$C_BISYNC_DELTA           =  0003D090
XG$C_BPC_8                  =  00000000
XG$C_BPC_DEFAULT            =  00000008
XG$C_BRG_19200              =  00000007
XG$C_BRG_DEFAULT            =  0000012C
XG$C_CIR_PAR                =  FFFFF700
XG$C_CTS_DELTA              =  000186A0
XG$C_DRPCTS                 =  00000006
XG$C_ERR_CRC1               =  00000000
XG$C_ERR_CRC16              =  00000003
XG$C_GENBYTE                =  00000007
XG$C_HEADER                 =  00000006
XG$C_IDLE                   =  00000000
XG$C_IND_ADDR               =  00000006
XG$C_INTCLK_OFF             =  00000000
XG$C_LINE_PAR               =  FFFFF300
XG$C_MISC_REG               =  00000004
XG$C_MODEM                  =  00000004
XG$C_NOCON                  =  00000007
XG$C_PRIRCV                 =  00000002
XG$C_PRIXMT                 =  00000000
XG$C_PRI_RCV                =  00000006
XG$C_PRI_RCV1               =  00000007
XG$C_PRI_XMT                =  0000000A
XG$C_PROTYPE_BISYNC         =  00000001
XG$C_PROTYPE_DDCMP          =  00000002
XG$C_PROTYPE_LAPB           =  00000000
XG$C_PRO_DDCMP              =  00000000
XG$C_PRO_HDLC               =  00000002
XG$C_RCV_CSR                =  00000000
XG$C_RCV_ERR                =  00000001
XG$C_SECRCV                 =  00000003
XG$C_SEC_RCV                =  00000008
XG$C_SEC_RCV1               =  00000009
```

```
XG$C_SIZEOFQ                    = 0000007F
XG$C_SWFCTS                     = 00000004
XG$C_SYNC_BISYNC                = 00000032
XG$C_SYNC_DDCMP                 = 00000096
XG$C_SYNC_HDLC                  = 00000000
XG$C_WFCTS                      = 00000002
XG$C_WFCTS_SEC                  = 00000006
XG$C_XMTING                     = 00000001
XG$C_XMT_CSR                    = 00000002
XG$C_XMT_ERR                    = 00000002
XG$DDT                            00000000 RG    03
XG$M_ABORT                      = 00000020
XG$M_BCC_ERR                    = 00000008
XG$M_BUFOVR                     = 00000040
XG$M_CTS                        = 00000010
XG$M_DONE_P                     = 00008000
XG$M_DONE_S                     = 00000080
XG$M_DTR                        = 00000200
XG$M_ENABLE                     = 00000001
XG$M_ERROR                      = 00004000
XG$M_ILP_XCS                    = 00000100
XG$M_INT_ENABLE                 = 00000020
XG$M_MASTER_RESET               = 00000080
XG$M_PRI_SEC_STN                = 00002000
XG$M_PRM_SEC                    = 00000004
XG$M_RES_BIT_CNT                = 00000700
XG$M_RTS                        = 00001000
XG$M_STRIP_SYNC                 = 00000040
XG$M_TERM_IDL                   = 00000008
XG$S_LOOP_TYPE                  = 00000002
XG$S_PROTOCOL                   = 00000003
XG$S_SYNC                       = 00000008
XG$S_XMT_BRG                    = 00000004
XG$V_BUFFER_CHAR                = 00000000
XG$V_BUFFER_IN_PREV_POS         = 00000001
XG$V_BUFOVR                     = 00000006
XG$V_CARRIER                    = 00000005
XG$V_COMPLETE_READ              = 00000002
XG$V_CTS                        = 00000004
XG$V_DONE_P                     = 0000000F
XG$V_DSR                        = 00000007
XG$V_DTR                        = 00000009
XG$V_ERROR                      = 0000000E
XG$V_LOOP_TYPE                  = 00000009
XG$V_MSG_LEN                    = 00000000
XG$V_PROTOCOL                   = 00000003
XG$V_RING_IND                   = 00000006
XG$V_RTS                        = 0000000C
XG$V_STRIP_SYNC                 = 00000006
XG$V_SYNC                       = 00000008
XG$V_XMT_BRG                    = 00000008
XG_DEF_BUFSIZ                   = 00000100
XG_DS_M_CLEAN                   = 00008000
XG_DS_M_CTSTQE_RUN              = 00004000
XG_DS_M_FORK_PEND               = 00000008
XG_DS_M_INITED                  = 00000004
XG_DS_M_LOADRPS                 = 00000020
XG_DS_M_RCVENB                  = 00001000
XG_DS_M_RCVING                  = 00000002
XG_DS_M_RCVPS                   = 00000010
XG_DS_M_RCV_DONEP               = 00000040
XG_DS_M_RCV_DONES               = 00000080
XG_DS_M_XMTING                  = 00000001
XG_DS_V_CLEAN                   = 0000000F
XG_DS_V_CTSTQE_RUN              = 0000000E
XG_DS_V_FORK_PEND               = 00000003
XG_DS_V_INITED                  = 00000002
XG_DS_V_LOADRPS                 = 00000005
XG_DS_V_RCVENB                  = 0000000C
XG_DS_V_RCVING                  = 00000001
XG_DS_V_RCVPS                   = 00000004
XG_DS_V_RCV_DONEP               = 00000006
XG_DS_V_RCV_DONES               = 00000007
XG_DS_V_XMTING                  = 00000000
XG_DS_V_XMT_DONEP               = 0000000A
XG_END                            0000257D R     03
XG_FC_V_READ_MODEM              = 00000005
XG_FC_V_STOP_CIR                = 00000002
XG_FC_V_STOP_LIN                = 00000003
XG_FC_V_STRT_CIR                = 00000000
XG_FC_V_STRT_LIN                = 00000001
XM$M_CHR_CTRL                   = 00000040
XM$M_CHR_DMC                    = 00000020
XM$M_CHR_HDPLX                  = 00000004
XM$M_CHR_TRIB                   = 00000080
XM$M_ERR_START                  = 00800000
XM$M_ERR_TRIB                   = 00400000
XM$M_STS_ACTIVE                 = 00000800
XM$V_CHR_HDPLX                  = 00000002
XM$V_CHR_LOOPB                  = 00000001
XM$V_CHR_MOP                    = 00000000
XM$V_CHR_TRIB                   = 00000007
XM$V_ERR_FATAL                  = 00000010
XM$V_ERR_LOST                   = 00000014
XM$V_ERR_MAINT                  = 00000013
XM$V_ERR_START                  = 00000017
XM$V_ERR_TRIB                   = 00000016
XM$V_MDM_CARRDET                = 00000000
XM$V_MDM_CTS                    = 00000002
XM$V_MDM_DSR                    = 00000003
XM$V_MDM_DTR                    = 00000006
XM$V_MDM_RING                   = 00000007
XM$V_MDM_RTS                    = 00000005
XM$V_STS_ACTIVE                 = 0000000B
XM$V_STS_BUFFAIL                = 0000000C
XM$V_STS_DISC                   = 0000000E
XM$V_STS_ORUN                   = 0000000A
XMITFDT                           00000229 R     03
XMITLAPB                          00000438 R     03
XMTQ$B_BUFTYP                     0000000A
XMTQ$B_FLAG                       0000001F
XMTQ$B_MSGHDR                     00000024
XMTQ$B_SLOT                       0000001C
XMTQ$C_LENGTH                     0000002A
```

```
XMTQ$K_LENGTH                     0000002A
XMTQ$L_BACC                       00000014
XMTQ$L_IRP                        0000000C
XMTQ$L_TIMEND                     00000010
XMTQ$M_INTERNAL                 = 00000080
XMTQ$M_ONQUEUE                  = 00000001
XMTQ$Q_LINK                       00000000
XMTQ$V_CONTROL                  = 00000002
XMTQ$V_INTERNAL                 = 00000007
XMTQ$V_SELECT                   = 00000001
XMTQ$W_BUFLEN                     00000008
XMTQ$W_DCRC                       00000022
XMTQ$W_ERROR                      0000001D
XMTQ$W_HCRC                       00000020
XMTQ$W_MSGOFF                     00000018
XMTQ$W_MSGSIZE                    0000001A
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000001C8 (456.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$105_PROLOGUE	0000006E (110.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	0000257D (9597.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	38	00:00:00.04	00:00:02.65
Command processing	145	00:00:00.59	00:00:04.22
Pass 1	1437	00:00:40.30	00:02:46.16
Symbol table sort	0	00:00:04.61	00:00:20.48
Pass 2	457	00:00:11.34	00:00:54.27
Symbol table output	2	00:00:00.45	00:00:01.55
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	2084	00:00:57.35	00:04:09.35

```
The working set limit was 3300 pages.
334316 bytes (653 pages) of virtual memory were used to buffer the intermediate code.
There were 240 pages of symbol table space allocated to hold 4051 non-local and 508 local symbols.
6434 source lines were read in Pass 1, producing 41 object records in Pass 2.
74 pages of virtual memory were used to define 68 macros.
```

```
                                                          I 14
```

Macro library statistics

Macro library name	Macros defined
_255DUA28:[DRIVER.OBJ]SYNCHLIB.MLB;1	9
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	33
_255DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	54

4248 GETS were required to define 54 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:XGDRIVER/OBJ=OBJ$:XGDRIVER MSRC$:XGDRIVER/UPDATE=(ENH$:XGDRIVER)+EXECML$/LIB+SHRLIB$:NMALIBRY/LIB+LIB$:SYNCHLIB/LIB
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

XGDRIVER
LIS

XIDRIVER
LIS